REVIEWER'S GUIDE FOR VMWARE THINAPP

VMware ThinApp 5.2

vmware®

Table of Contents

Introduction

Welcome to the *Reviewer's Guide for VMware ThinApp® 5.2*. The purpose of this guide is to familiarize you with VMware ThinApp, as well as with features introduced since ThinApp 4.x. Practical exercises help you evaluate both new and ongoing key features.

Audience

This guide is for anyone who wants to install ThinApp and deploy captured applications. Typical users are system administrators responsible for the distribution and maintenance of corporate software packages. Both current and new users of ThinApp can benefit from using this guide.

Objectives: What You Will Learn

This guide introduces you to ThinApp and gives you practical exercises to evaluate the product. Following is the overall organization of the guide.

- What is VMware ThinApp?
- Ongoing key features
- Licensing
- Architecture and components
- Practical exercises

Navigating This Document for VMware ThinApp Use Cases

You can navigate directly to the discussion of essential ThinApp features, some of which include accompanying practical exercises.

- ThinApp Packaging Process
- Deploy ThinApp Packages
- Update ThinApp Packages
- Use AppLink to Combine ThinApp Packages
- Use ThinApp Packages with App Volumes
- Use ThinApp Packages with Mirage
- Use ThinApp Packages with View
- Use ThinApp Packages with VMware Identity Manager™

What Is VMware ThinApp?

ThinApp virtualizes applications by encapsulating their files and registry settings into a ThinApp *package*, which includes one or more executable files that can be started without requiring installation. IT administrators can deploy, manage, and update these ThinApp packages independently from the underlying operating system. The virtualized applications do not make any changes to the underlying operating system and behave the same across different desktop configurations. They are also isolated from each other and from the underlying operating system. This provides a stable and consistent end-user experience.

Some use cases for ThinApp include delivering an application through removable media to remote users with laptops, or running different versions of the same application concurrently. ThinApp can be used to run legacy applications on newer operating systems.

How ThinApp Complements Other VMware Products

Although ThinApp is available as a standalone product, it is an integral part of the VMware End-User Computing vision. ThinApp is now bundled in each VMware Horizon 7® edition, adding enhanced features and advantages to VMware App Volumes™, View in VMware Horizon 7, VMware Mirage™, and VMware Identity Manager (formerly VMware Workspace™ Portal).

As an essential component of the Horizon 7 editions, ThinApp adds application compatibility to the virtual workspace and helps reduce the management burden of provisioning, patching, and updating applications and images.

PRODUCT OR COMPONENT	HORIZON 7 STANDARD EDITION	HORIZON 7 ADVANCED EDITION	HORIZON 7 ENTERPRISE EDITION
ThinApp	✔	✔	✔
App Volumes			✔
Mirage		✔	✔
VMware Identity Manager		✔	✔
View in Horizon 7	✔	✔	✔

TABLE 1: Horizon 7 Edition Products and Components

ThinApp creates virtual application packages to deploy to View virtual desktops, to deploy to Mirage-managed desktops, to include in the VMware Identity Manager application catalog, or to deploy directly to physical or virtual machines. ThinApp packages can also be delivered through VMware App Volumes on VMDKs, or streamed across the network from a file share.

When should you integrate ThinApp with other VMware products? As individual requirements vary from environment to environment, there is no definitive answer. Based on your requirements, you can choose whether or not to use ThinApp with some or all of these products. The following sections discuss the benefits that ThinApp can bring to each product, the drawbacks, and the options to alleviate those drawbacks. Table 2 summarizes this information.

PRODUCT OR COMPONENT	WORKS WITH OFFLINE ENDPOINTS	FREQUENT ONLINE UPDATES	USES NETWORK BANDWIDTH	USES STORAGE
App Volumes		✔		✔ (VMDKs)
Mirage	✔			✔ (local)
VMware Identity Manager		✔	✔	✔ (local)
View in Horizon 7		✔	✔	✔ (local)

TABLE 2: Summary Considerations for Integration with ThinApp

App Volumes and ThinApp

App Volumes provides near-instant application delivery to end-user desktops. App Volumes stores applications in shared read-only virtual disks (VMDK files called *AppStacks*) that are assigned to users, groups, or computers.

Benefits

With App Volumes, administrators can deliver ThinApp packages with AppStacks. ThinApp provides exceptional application isolation capabilities. Using App Volumes to deliver ThinApp packages provides the best of both worlds—real-time delivery of isolated and legacy applications alongside natively delivered applications.

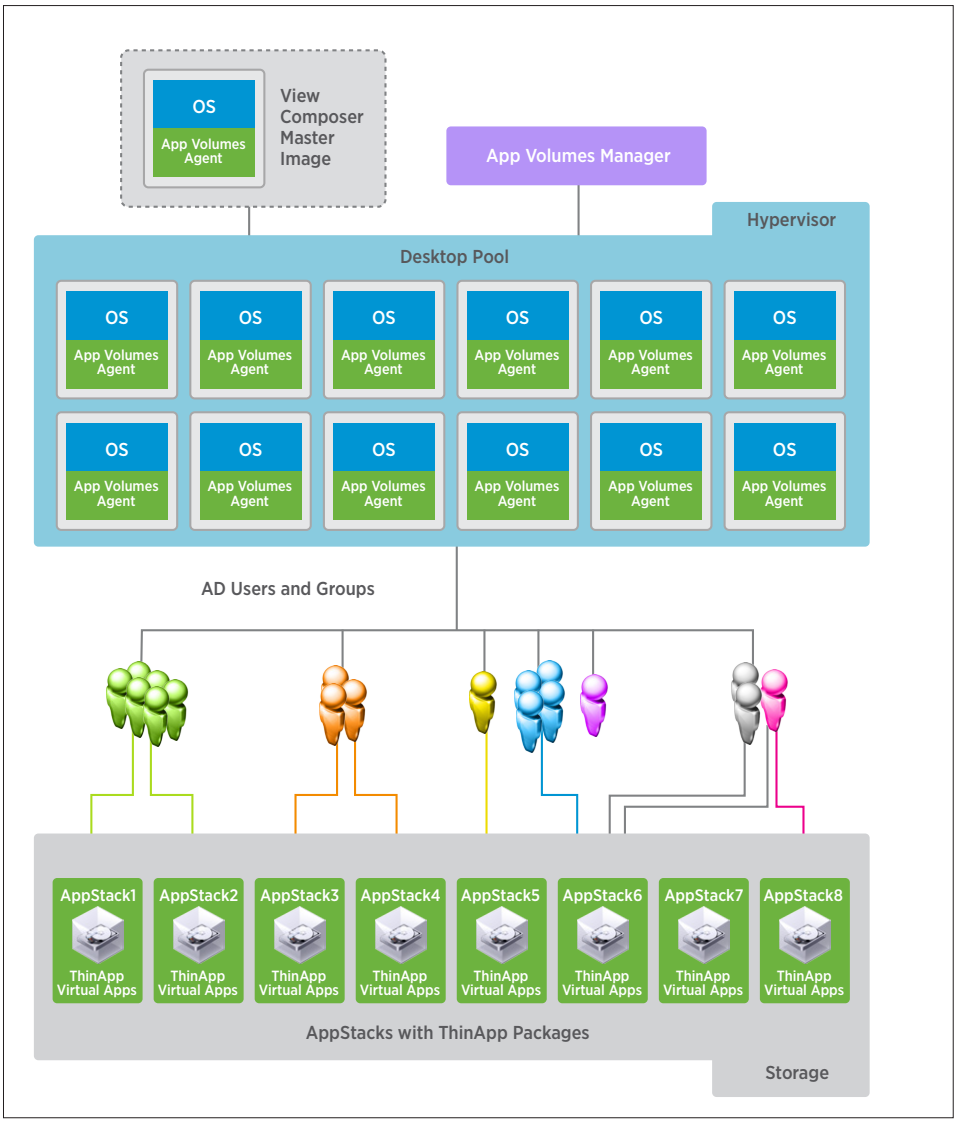

FIGURE 1: ThinApp Packages in AppStacks

Deploying ThinApp packages with App Volumes provides the following benefits.

- **Improved enterprise entitlement** – User-based entitlement is not a feature of ThinApp. However, App Volumes does allow for flexible entitlement of AppStacks to Active Directory (AD) users, groups, and computers. When you integrate ThinApp with View in Horizon 7 and App Volumes, you can entitle ThinApp packages to desktops and pools, as well as to AD users, groups, and computers.
- **Near-instant application delivery** – App Volumes delivers applications almost instantly. When an AppStack is attached to a virtual desktop, the user sees the assigned applications almost literally at the click of a button. You can store AppStacks on any supported VMware vSphere® datastore, so IT can leverage the most efficient storage in the environment. In addition, you can assign a single AppStack to many desktops, which contributes to the rapid delivery of applications in an enterprise environment while helping reduce storage requirements.
- **Enhanced performance** – In comparison with the current View and ThinApp streaming integration, App Volumes does not move data across the network. Although ThinApp streaming has its advantages, it does require some design considerations. ThinApp streaming is dependent on network bandwidth, while App Volumes delivery and execution relies only on enterprise storage performance. Using App Volumes to deliver ThinApp packages results in performance that is predictable at scale. You get the performance of a locally installed ThinApp package rather than the performance of a streamed ThinApp package that is accessed from a network file share.
- **ThinApp package reuse** – You do not need to recapture applications to use them with App Volumes. You can easily reuse existing ThinApp packages with App Volumes. Deploying these packages with App Volumes provides all the benefits of local deployment with centralized administration.

Drawbacks

App Volumes does not deliver applications to offline endpoints. If you have desktops and laptops with intermittent network connection, you can use Mirage to deliver ThinApp packages to those endpoints.

View and ThinApp

View simplifies and automates the management of desktops, and securely delivers them as a service from a central location to users throughout the enterprise and at remote locations. Users get a familiar, personalized environment that they can access from any number of endpoints.

Benefits

Using ThinApp packages on virtual desktops offers many benefits. The combination of View and ThinApp simplifies desktop management by streamlining application delivery, eliminating application conflicts, and allowing for updates of applications without end-user disruption.

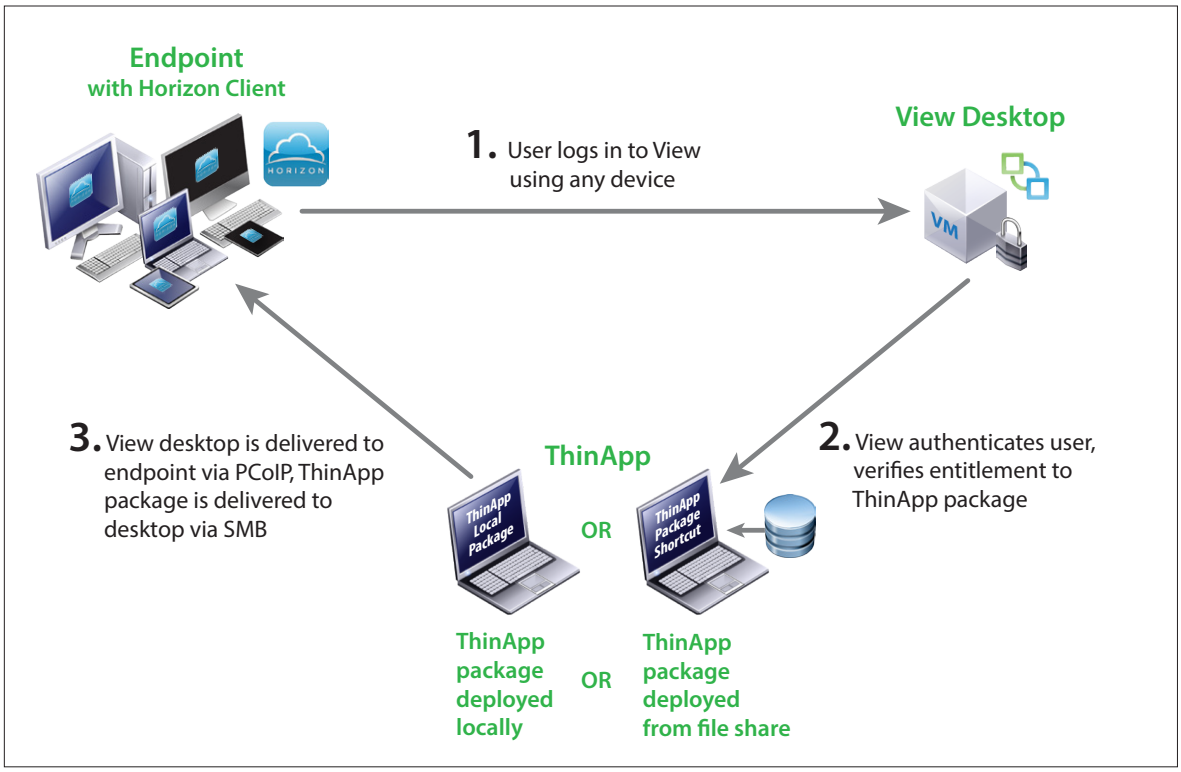

FIGURE 2: Delivering ThinApp Packages Through View Alone

Using ThinApp with View desktops in Horizon 7 includes the following benefits.

- Application virtualization eliminates the need to install an application on every desktop. You can place a single ThinApp package on a file share, and users can independently stream the application to their desktops, reducing the cost of supporting individual copies of the application on multiple desktops.
- Application maintenance is simplified because you can upgrade and patch a single instance of an application and deliver it to many users.
- Virtualized applications can be used with persistent or nonpersistent desktops. Administrators can use folder redirection for the ThinApp application sandbox, which allows users to save their customized application settings, such as toolbar and option settings, to a network location. Users get persistent application settings regardless of whether the desktop is persistent or nonpersistent.

Drawbacks

ThinApp packages can be stored locally on the endpoints or on a network share.

Packages run from a network share use network streaming. Streaming relies on network bandwidth and this can negatively impact performance. To counteract this, you have two options.

1. Use App Volumes which does not move data across the network.
2. Deliver the ThinApp package directly to the local endpoints.

If the ThinApp package is delivered to a local endpoint, it does not use network bandwidth.

Mirage and ThinApp

VMware Mirage is a layered-image management solution that separates a desktop, laptop, or other endpoint into logical divisions called *layers*. Layers are owned and managed by either your IT organization or the end user, and are useful for creating standardized desktop configurations.

Benefits

ThinApp can take advantage of the Mirage layering technology because the ThinApp package is encapsulated. Organizations with existing ThinApp packages can use Mirage as a simple method of package deployment and management, providing ThinApp packages in base or application layers.

Using ThinApp together with Mirage gives you the capability to address every application on the Windows desktop. If you need applications with drivers and remote DCOM, you can install these native applications on Mirage application layers. If you need cross-platform support or isolation, you can install ThinApp packages on Mirage application layers. Administrators can deploy Mirage layers and ThinApp packages and centrally manage both native and virtualized applications.

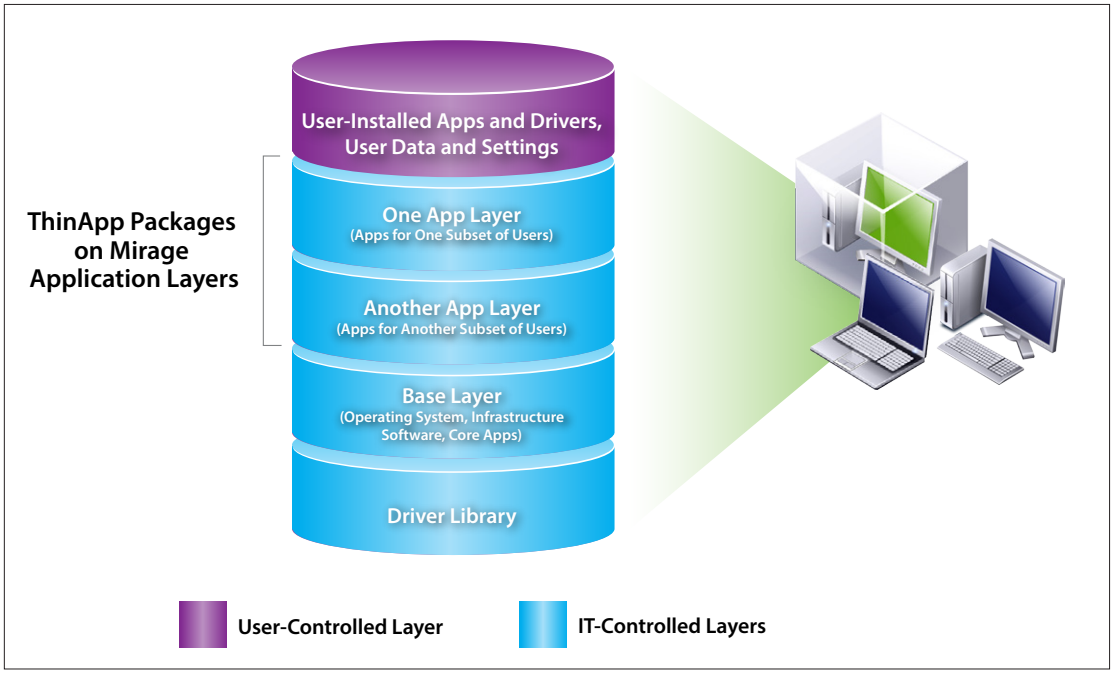

FIGURE 3: Delivering ThinApp Packages Using Mirage Layers

Drawbacks

Mirage delivers applications to physical Windows systems that are not always online. If the majority of your endpoints are virtual machines, you can deliver ThinApp packages using View, App Volumes, or VMware Identity Manager.

VMware Identity Manager and ThinApp

VMware Identity Manager simplifies the end-user experience and reduces IT costs by combining applications, data, and desktops into a single, enterprise-class *catalog* securely delivered to any endpoint.

VMware Identity Manager is another platform in Horizon 7 that you can use to deliver ThinApp packages to users.

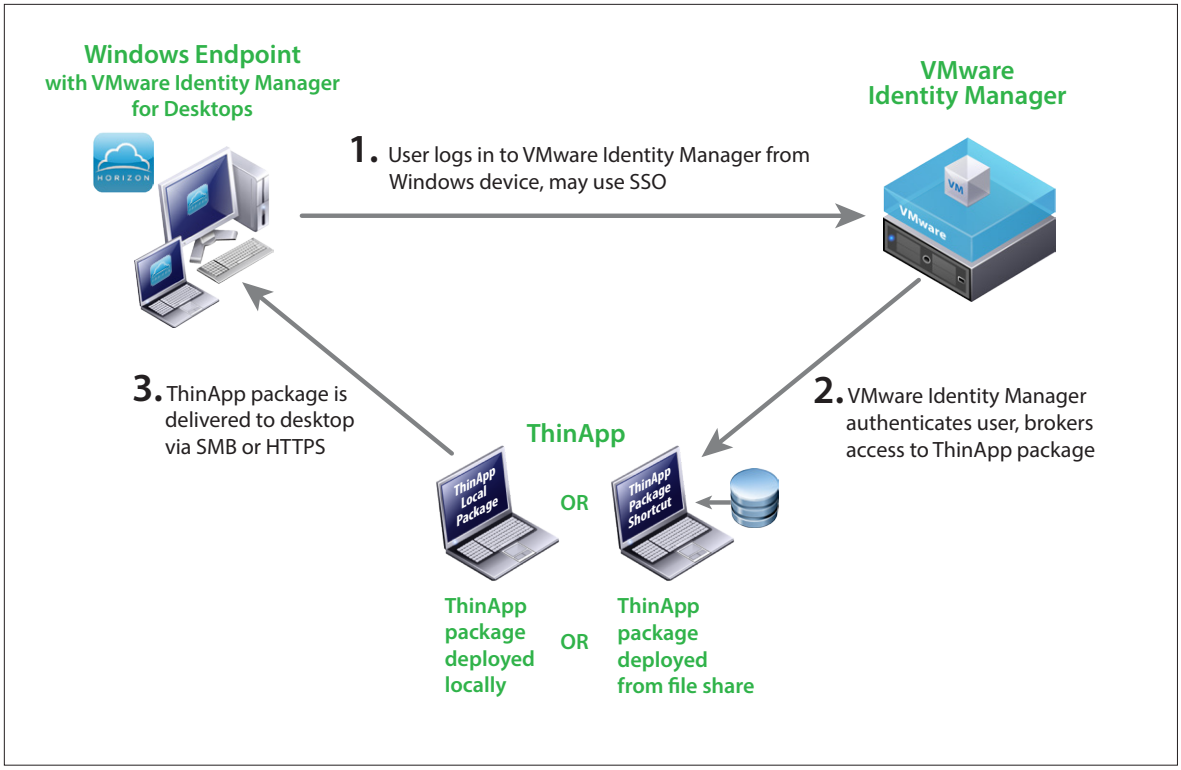

FIGURE 4: Delivering ThinApp Packages Through VMware Identity Manager

Benefits

The main benefit of delivering ThinApp packages through VMware Identity Manager is the simplified delivery mechanism. Users have a customized application portal tailored to their specific application requirements. You can assign ThinApp packages to a specific person or group of people. Because you entitle users rather than desktops, a user can run the ThinApp package from any physical or virtual Windows machine.

Drawbacks

ThinApp packages managed by VMware Identity Manager can be stored on the local endpoint or on a network share.

Packages run from a network share use network streaming. This means that performance is dependant on network bandwidth. To counteract this, you have two options.

1. Use App Volumes which relies on storage performance as opposed to network performance.
2. Deliver the ThinApp package directly to the local endpoints.

If the ThinApp package is delivered to a local endpoint, it does not use network bandwidth.

ThinApp Use Cases

Common use cases for ThinApp are

- **Simplify using legacy applications and Web browsers on newer operating systems** – Migrate legacy applications and Web browsers to newer operating systems by packaging them with ThinApp.
- **Eliminate application conflicts** – Isolate desktop applications from each other and from the underlying OS to avoid conflicts.
- **Consolidate session-based servers** – Minimize application conflicts on servers, and as a result, reduce the number of servers required to host the applications.
- **Improve application security and life cycle management** – Add ThinApp packages to View desktops, and centrally manage user access and application updates.
- **Increase mobility for end users** – Deploy, maintain, and update virtualized applications on removable media for increased portability.
- **Support bring-your-own-PC (BYOPC) policies** – Deploy virtualized applications to user-managed endpoints with VMware Identity Manager.

New Features

There have been several new features since ThinApp 4.x.

Note: For more detailed discussion, see New ThinApp Features.

New Features in ThinApp 5.2.x

ThinApp 5.2 and 5.2.1 were maintenance releases, but ThinApp 5.2 also introduced support for Windows 10.

New Features in ThinApp 5.1

ThinApp 5.1 introduced the following features.

- Active Directory group policies override of AppLink and AppSync
- Enhanced ThinDirect including
  - Redirection between virtual instances
  - Mozilla Firefox support
  - Update of ThinDirect settings at specified time intervals
  - Dynamic changes to ThinDirect using Active Directory group policies
- Project to physical (P2P)
- MAPI support
- Support for Internet Explorer 10 and 11
- Support for Windows 8.1

New Features in ThinApp 5.0.x

ThinApp 5.0 and 5.0.1 introduced the following features.

- A Setup Capture wizard link to the VMware knowledge base article Guidelines on virtualizing applications with ThinApp (2070839)
- Support for 64-bit SDK
- New `Package.ini` parameters
- Support for 64-bit applications
- Support for a 64-bit virtual command prompt

Ongoing Key Features

There are several ongoing key features of ThinApp.

- Standalone-executable application virtualization
- Easy, flexible application packaging
- Fast, flexible application delivery
- Seamless integration with existing VMware products
- Seamless integration with third-party products

Details of these features follow.

Standalone-Executable Application Virtualization

- **Application encapsulation** – ThinApp is designed for fast deployment and ease of management. It requires no installation, and leaves no code or other software components on target endpoints. Applications are encapsulated into a single executable file.
- **Complete application isolation** – Encapsulated applications and their settings run independently on any endpoint that supports them, allowing multiple versions of an application, or multiple applications, to run on the same endpoint without conflict.

Easy, Flexible Application Packaging

- **Package once, deploy to many** – Package an application once and deploy it to many desktops and servers (physical and virtual, 32-bit and 64-bit) running many different versions of Windows.
- **Easy Application Packaging** – Use a process that scans pre- and post-install system states to simplify application packaging and to support applications that require a reboot during the installation process.
- **Internet Explorer support** – Easily virtualize and deploy different Internet Explorer versions to the same Windows desktops.
- **Relink** – Use this ThinApp utility to quickly upgrade existing ThinApp packages to a newer version of ThinApp without having to rebuild them from their projects.

Fast, Flexible Application Delivery

- **ThinDirect** – An administrator can configure specific Web pages to open in specific versions of a Web browser. This ensures that URLs always open in the right browser.
- **Application Link (AppLink)** – Link one or more ThinApp packages together into a single virtual environment. This allows you to configure relationships between virtualized applications, plug-ins, service packs, and even runtime environments such as Java and .NET.
- **Application Sync (AppSync)** – Automatically apply updates over the Internet to applications running on unmanaged endpoints.
- **Support for USB removable media and thin clients** – Deploy, maintain, and update applications on USB removable media and thin clients.
- **Support for legacy applications and Web browsers** – Virtualize legacy applications and the Web browsers they require, making them easily available to newer Windows operating systems.

Seamless Integration with Existing VMware Products

- **App Volumes** – Deliver ThinApp packages to desktops almost instantly and assign them to Active Directory users, groups, and computers.
- **Integrated application assignment in View** – Assign ThinApp packages to individual desktops or pools of desktops in View to allow for streamlined application deployment.
- **Mirage** – Deploy existing ThinApp packages to endpoints that include physical machines, virtual machines, and mobile devices.
- **VMware Identity Manager** – Deliver ThinApp packages, alongside Web, SaaS, and Windows applications, allowing a seamless end-user experience across multiple endpoints.

Seamless Integration with Third-Party Products

- **Integration with management tools** – Deliver ThinApp packages using existing application deployment tools from BMC, CA, HP, LANDesk, Microsoft, Novell, Symantec, and others.
- **Support for Active Directory authentication** – Add and remove ThinApp users from Active Directory groups, and prevent unauthorized users from executing ThinApp packages.

Obtaining ThinApp

You can purchase ThinApp as part of VMware ThinApp Suite, VMware Identity Manager, VMware Mirage, or as part of the Horizon 7 editions. You can also obtain an Evaluation license.

For more information, see Appendix A.

Supported Platforms, Applications, and Systems

You can run ThinApp packages on the following operating systems and server configurations.

- 32-bit platforms – Windows 2000, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Windows Server 2008
- 64-bit platforms – Windows 2000, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2
- RDSH server and Citrix XenApp

Note: Microsoft no longer supports Windows XP or Windows 2003, so these operating systems are no longer supported as ThinApp deployment platforms. However, Windows XP is still supported as an operating system for ThinApp capture machines only.

ThinApp supports virtualizing the following application architectures.

- 16-bit applications running on 32-bit Windows operating systems
- 32-bit applications running on 32-bit and 64-bit Windows operating systems
- 64-bit applications running on 64-bit Windows operating systems

Not Supported

- Virtual applications running on 16-bit or non-x86 operating systems, such as Windows CE
- 64-bit virtual applications running on 32-bit Windows operating systems
- 16-bit virtual applications running on 64-bit Windows operating systems
- Applications requiring installation of kernel-mode device drivers (ODBC drivers are supported because they are user-mode device drivers)
- Products such as antivirus and personal firewalls
- Scanner and printer drivers and some VPN clients

Architecture and Components

This section describes how ThinApp and its components and architecture work.

How ThinApp Works

ThinApp uses a build process to package application files and registry settings into a single application container that can be executed on a variety of operating systems without installation. The application container uses block-based streaming with transparent decompression into memory to execute all application functions. Applications can be executed from a user's desktop, a network path, or removable media. Applications run entirely in user mode under the security context of the currently logged in user. ThinApp presents operating system resources and functions to the virtualized application, providing a seamless user experience while encapsulating the application's files, registry entries, COM/ActiveX controls, and services in a portable container for use across multiple operating systems.

The process of virtualizing an application with ThinApp begins with the Setup Capture tool and ends with building a read-only redistributable package that encapsulates all the necessary components of the application along with the administrator-determined configuration settings necessary for implementation. Setup Capture creates a project folder to store the application files and configuration settings. The build process embeds these application files and configuration settings into a package. The project folder is the source location where the administrator can update and change the package configuration. The process of using Setup Capture, modifying project directories, and building packages is iterative, and often referred to as *capture and build*.

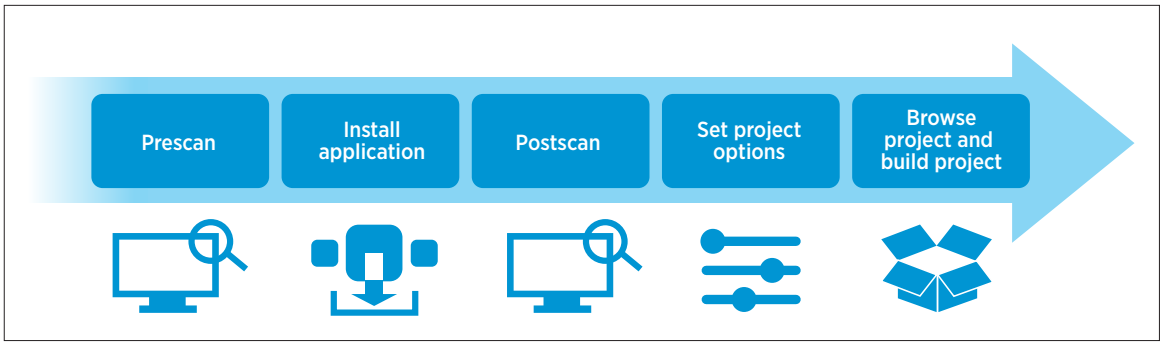

FIGURE 5: Capture and Build Process

For information about Setup Capture, see the ThinApp documentation.

ThinApp Architecture

ThinApp simplifies application delivery by isolating applications from the underlying operating system.

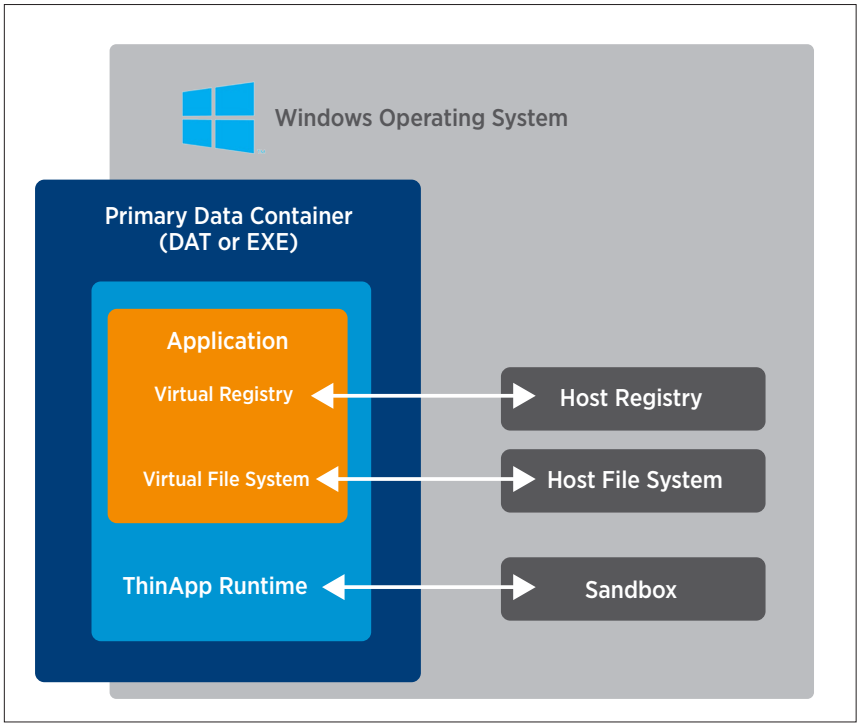

FIGURE 6: ThinApp Architecture

The ThinApp runtime is encapsulated in its own container, called the Primary Data Container or PDC. The PDC is the main data file for the virtualized application and holds the ThinApp runtime and its virtual registry and virtual file system. Each application is isolated from all other applications and the operating system.

When the ThinApp runtime is started, it controls which elements are written to the sandbox and which are written to the host system. It also controls the processes and threads created in the operating system, including out-of-process COM and child processes.

Communication between the virtual registry or virtual file system and the host registry or host file system depends on the isolation mode. Isolation modes allow you to control the degree to which a virtualized application can read from and write to the local-host PC where the virtual application resides. ThinApp project settings determine the file system and registry key isolation modes.

The default file system isolation mode for a ThinApp project is Merged, which enables users to create and update files in non-system directories, such as `Desktop` and `Documents`. Changes to files in system directories are stored in the sandbox. During Setup Capture, you can set your project to WriteCopy isolation mode. In this case, new and updated files, except files in the user's `Desktop` and `My Documents` directories, are stored in the sandbox. The default registry isolation mode is always WriteCopy, regardless of which mode you selected during Setup Capture. However, you can change the registry isolation mode by adding the `RegistryIsolationMode` parameter to `Package.ini` and rebuilding.

For example:

```
[Isolation]

RegistryIsolationMode = Merged
```

If the application is isolated from the host system, all changes, deletions, and additions made by the application to the file system or registry are recorded in the sandbox instead of the host operating system. The default sandbox location is `%APPDATA%\Thinstall`. However, you can store the sandbox in any location (for example, on a network share) if the users have modify permissions. Exclude the sandbox from antivirus scanning because it can corrupt the sandbox and cause the virtual application to fail.

For more information about registry modes and the sandbox, see the VMware knowledge base article Understanding the ThinApp sandbox and isolation modes (1030224).

File Share Considerations

Standard Windows file share requirements apply to file shares used for ThinApp packages in remote deployments.

Size your ThinApp file share the same way as you would a traditional file share that provides access to Excel and Word documents. To protect the content of the file share, limit users to read and execute permissions.

It is recommended to exclude the file share from antivirus scanning, specifically on-access scanning, because it negatively impacts performance.

Security Considerations

The first layer of security is within your network infrastructure. The next resides on your endpoint.

You can add an extra layer of security to your ThinApp deployment by using isolation modes. Isolation modes prevent users from making changes to the native OS file system or registry.

The three isolation mode settings are Merged, WriteCopy, and Full. If you set the isolation mode to Full, the virtual application cannot read host data, and any write operations are directed to the sandbox. For more information about the isolation mode settings, see *Isolation modes* in ThinApp Components and Terminology.

Another way to control application access is by using Active Directory groups. You can select this option during Setup Capture or edit the `PermittedGroups` parameter in the `Package.ini` file. You can restrict access to a specific group of users to further protect ThinApp packages from unauthorized use.

AppSync can be used to control application access. Enable the `AppSyncExpirePeriod` parameter in `Package.ini` to expire the application after a specified time period. The default value is 30 days. For example, this feature is useful in a scenario where a contractor needs temporary access to applications.

You can also prevent the application from starting if the user is not logged in to the network. Set the following parameters in `Package.ini`:

```
AppSyncUpdateFrequency=0

AppSyncExpirePeriod=0d
```

This forces the application to check for an update every time it is started.

For more information, see the ThinApp Package.ini Parameters Reference Guide.

VMware Identity Manager offers advanced security and protection of corporate applications. Administrators can protect ThinApp packages by restricting access to authorized users and groups. Enable VMware Identity Manager management during Setup Capture or enable `AppID` and `NotificationDLLS` in the `Package.ini` file. Virtual applications managed by VMware Identity Manager cannot start without valid entitlement from VMware Identity Manager.

ThinApp Components and Terminology

The ThinApp lifecycle is split into three main phases. The first phase, capture and build, involves using Setup Capture, modifying project directories, and building packages. The next phase deals with the distribution of ThinApp packages and is referred to as *deployment*. The final phase, which updates and upgrades ThinApp packages, is an *update* process.

Capture and Build Terminology

The capture and build process uses the following terminology.

Setup Capture – The Setup Capture wizard guides the process of capturing the application and applying administrator-supplied configurations specific to the package. Setup Capture takes a *prescan* snapshot, allows the administrator to install and configure the application, and takes a *postscan* snapshot. The difference between the prescan and postscan snapshots, which represents the application, is placed in the project directory. Setup Capture supports traditional installers to install applications, basic file copies, and manual environment changes. You can reboot the capture machine any number of times before taking a postscan snapshot.

Project – The output of Setup Capture is a project. A project contains everything needed to build and modify the virtualized application. The project folder contains the following:

- Directories and files that have been changed by the application's installation procedure. These directories are listed by the common Windows folder macros, such as `%AppData%` and `%SystemRoot%`.
- Directories that represent the common Windows folders contain an `##Attributes.ini` file, which contains isolation mode settings for the particular location.
- Registry changes made by the application and the isolation mode settings are contained in TXT files named `HKEY_Current_User.txt`, `HKEY_Local_Machine.txt`, and `HKEY_Users.txt`.
- The `bin` directory contains the package files in `.exe` or `.msi` format created from the build process.
- The `Support` directory contains the `Capture Machine Overview.txt` file, which records the post-packaging specifics of the system. This file is helpful for troubleshooting.
- The `Package.ini` file contains the settings recorded by Setup Capture and other project-wide configuration settings. The `Package.ini` file is a repository of all ThinApp configuration data for deployment, updates, shortcuts, and entry points of that particular application package.
- The `build.bat` file initiates the build process by making calls to three components in the ThinApp installation folder, which in turn compile the virtual registry, virtual file system, and application logic into the executable file. You can double-click this file to create the ThinApp-packaged executable.

Build – The process by which the project directories and configuration settings are compressed and embedded into the package. The build operation is the last step in the Setup Capture wizard. You can also manually build a project by using the `build.bat` file in the project directory. You can rerun the build process at any time to incorporate different settings or application changes into a package. Any machine with access to the file share that contains the project directory, and which can find the ThinApp executables `vftool.exe`, `vfregtool.exe`, and `tlink.exe` in its path, can build the package. You do not need to return to the source machine that created the package, be on a similar operating system, or use a management console to build the package.

Package – The capture and build process creates a package—an application container that encapsulates the application, the ThinApp runtime, and all the required configuration settings. A package is created in a ready-to-run, read-only EXE format. Additionally, the package can be placed in an MSI wrapper for native integration with software deployment tools. The MSI wrapper includes the packaged EXE file, the Thinreg tool, and the MSI database condensed into one file for ease of deployment. The package is transportable and requires a supported operating system to run the application. ThinApp packages are most often named after the application, such as `Mozilla Firefox.exe` or `Mozilla Firefox.msi`.

Isolation modes – The isolation mode specifies the degree of isolation between the virtualized application and the files, folders, and registry of the local operating system. By default, isolation mode settings are inherited through a folder and registry structure. Administrators have granular control to define the isolation mode setting for each folder and registry subtree. The three settings are Merged, WriteCopy, and Full.

- **Merged** allows you to interact with the physical environment. Applications can read and create new elements on the host file system outside of the virtual package.
- **WriteCopy** allows ThinApp to intercept write operations and redirect them to the sandbox. You can protect your host environment from user changes, because any changes made are saved in the sandbox. The virtual application can read host data, but cannot change it.
- **Full** prevents the ThinApp application from accessing locally available data in any way. All write operations happen in the sandbox, and host data cannot be read.

For more information, see the VMware knowledge base article Configuring isolation modes for the file system and registry in ThinApp (1017265).

Sandbox – A user-specific folder created for each virtualized application that holds runtime changes to the virtualized registry, folders, and files. The virtualized application maintains the sandbox and uses isolation mode settings to determine when to write changes to the sandbox or to the local operating system. The sandbox is located by default in the user's `%AppData%` location. If folder redirection or roaming profiles are in use, the sandbox follows that path. The sandbox folder location is configurable and can be located centrally for users to access their application settings from multiple endpoints.

Entry points – User-accessible starting points for virtualized applications or natively installed applications for use by virtualized applications. By default, entry points correspond to the executables detected during Setup Capture and chosen by the administrator as the entry point for the end user into the virtualized application.

Deployment Terminology

The deployment process uses the following terminology.

Deployment mode – Two options for providing virtualized applications to end users are available.

- **Remote deployment** allows the application to be centrally stored and accessed by multiple users. Remote deployment is a one-to-many model that provides centralized deployment and updating of an application package to multiple end users for execution through a Windows desktop shortcut. Remote deployment requires only a file share.
- **Local deployment** application packages are first deployed to the end user's system and accessed from the local endpoint. Users execute the application from a local application package, which allows for offline application use.

Thinreg.exe – This utility automates the registration of application shortcuts on the desktop and the Start menu, file-type associations, and entries in the Programs and Features applet of the Control Panel. You can run Thinreg from a script, the command line, or a login script. It is also incorporated into MSI packages when this feature is selected within the ThinApp build process.

Update Terminology

The update process uses the following terminology.

AppLink – This feature connects one or more ThinApp packages at runtime. The administrator can build relationships between projects to package, deploy, and update component pieces in separate ThinApp packages rather than use Setup Capture to package all needed components into a single executable.

AppSync – This setting initiates the poll of an updated package from a central Web server or UNC location. The interval for polling for updates and the location of the HTTP service or file share are configurable, along with other settings in the `Package.ini` file.

Sandbox merge – The SBMerge utility allows the administrator to merge runtime changes from an existing sandbox into the project directory of a captured application. The virtualized application package can then be rebuilt and distributed to end users, incorporating the changes from the merge process.

Side-by-side update – This method places a new application package in the same directory as the original application package, and uses an integer extension in its filename to distinguish it. Subsequent updates can be placed in the directory with incremented extensions (2, 3, and so on). This method is also known as an integer update or in-place update. When multiple filenames are found, the one with the highest integer extension will be run. For a practical exercise, see Side-by-Side Update.

New ThinApp Features

ThinApp includes the following new features since ThinApp 4.x.

Package Management

To change project settings in previous ThinApp versions, you had to edit `Package.ini` parameters, save the file, and rebuild the project. With the new package management feature, administrators can now dynamically reconfigure certain attributes of deployed ThinApp packages at runtime. After the initial configuration is complete, administrators can easily manage and apply project changes to packages within minutes using Active Directory group policies.

ThinApp 5.1 introduces Group Policy Administrative Template (ADMX and ADML) files, which allow administrators to reconfigure group policy settings for ThinApp applications.

The group policy files work on domain controllers running in the following environments.

- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2

You can reconfigure the attributes of the following aspects of a deployed package.

- AppSync
- AppLink
- Entry-point shortcuts
- ThinDirect

After ThinApp 5.1 has been installed, a folder named `Policy` is created in the installation directory or the ThinApp utilities folder. The `Policy` folder contains the following tools and templates.

- `AppPolicy.exe`
- `ThinAppBase.adml`
- `ThinAppBase.admx`
- `ThinAppGeneric.adml`
- `ThinAppGeneric.admx`

AppPolicy

You must create the GPO template files for each ThinApp package that you need to manage. The AppPolicy utility creates and customizes the generic template files of the application that you want to reconfigure or manage. However, you must use the exact inventory name.

To get the correct inventory name, run the following from a Windows command prompt:

```
AppPolicy.exe /s <path to PDC>
```

Note: The generic template files must be in the same directory as `AppPolicy.exe`. For more information, see *ThinApp Package Management* in the ThinApp User's Guide.

Template Files

The administrative template files contain markup language that is used to describe registry-based group policies. The administrative files are divided into language-neutral (ADMX files) and language-specific (ADML files) resources. This allows group policy tools to adjust the Group Policy Editor UI according to the administrator's configured language.

The `ThinAppGeneric.admx` and `ThinAppGeneric.adml` files are used by AppPolicy to create ThinApp group policy templates.

The `ThinAppBase.admx` and `ThinAppBase.adml` files provide additional information to ThinApp group policy objects, including their *base namespace* and *category structure*.

Enhanced ThinDirect

ThinDirect has been enhanced in ThinApp 5.1 with several new features.

Architecture redirection – ThinDirect now supports redirection between 32-bit and 64-bit browsers.

Virtual-to-virtual-browser redirection – In earlier ThinApp versions, ThinDirect entries for virtual-to-virtual-browser redirection were only checked when the native browser was started, resulting in the native IE browser being automatically started (and subsequently closed) when the user accessed a redirected page from a virtual browser. This scenario confused users because multiple browser windows were started.

The new ThinDirect feature now supports redirection directly between two virtual browser instances, allowing users to specify URLs in different virtualized browsers without the native browser being started.

Mozilla Firefox support – Previous versions of ThinDirect allowed redirection from native Internet Explorer to virtual Firefox. However, redirection from native Firefox to a virtual browser was not possible.

ThinApp 5.1 introduced full ThinDirect support for Firefox version 22 and later, by making ThinDirect available as a plug-in for Firefox. Administrators can redirect pages from a Firefox browser to another browser. Native IE is no longer the required default browser.

Dynamic update of ThinDirect settings – ThinDirect now periodically polls for changes to its settings and dynamically updates browser instances that are already running. As a result, users do not need to restart their browsers for the changes to take effect.

GPO Override of AppLink, AppSync, Entry-Point Shortcuts, and ThinDirect

With ThinApp 5.1, administrators can use group policies to override `Package.ini` settings for AppLink, AppSync, entry-point shortcuts, and ThinDirect.

If a GPO is configured, the GPO settings take precedence over the settings in the `Package.ini` file. For AppLink, AppSync, and entry-point shortcuts, use the `ThinAppGeneric.admx`, `ThinAppGeneric.adml`, `ThinAppBase.admx`, and `ThinAppBase.adml` files to create ThinApp group policy templates for specific applications.

Use the `Thindirect.admx` and `ThinDirect.adml` files to manage ThinDirect settings using GPOs and override the `ThinDirect.txt` file.

For a practical exercise on overriding AppLink settings, see Configure AppLink Using Group Policy Objects.

Project to Physical (P2P)

You can now use command-line options to extract an existing ThinApp project into a capture and build environment. This means that rather than installing a native application after taking a prescan, you can use P2P to "install" the application from its existing project and make further modifications to it before taking a postscan and creating a new project based on those modifications. This saves time and effort in replicating a previously captured project that you need to modify.

P2P uses the `snapshot.exe` and `snapshot64.exe` commands to extract the project. The user running P2P must be the same user who captured the project, and P2P must be run from the same OS version the project was captured on.

For more information, see the VMware blog post The New ThinApp 5.1 Feature "Project to Physical" Explained.

MAPI Support

ThinApp 5.1 supports the Messaging Application Programming Interface (MAPI) on 32-bit and 64-bit versions of Windows 7, Windows 8, and Windows 8.1.

ThinApp 5.1 provides the `DefaultEmailProgram` option in `Package.ini` to register a virtual email client as the host's default email program. This option must be enabled to register the email program.

Support for Internet Explorer 10 and 11

ThinApp 5.1 supports Internet Explorer 10 and 11 only on Windows 7.

For more information about supported Internet Explorer and Windows versions, see the VMware knowledge base article Support Policy for Internet Explorer virtualized with VMware ThinApp (2069870).

Support for Windows 8.1 and Windows 10

ThinApp 5.1 runs on the Windows 8.1 August update (Update 2), and ThinApp 5.2 works on Windows 10.

Package.ini Updates

`Package.ini` parameters are now case insensitive. The following parameters are now present by default in the `Package.ini` file.

- `ChildProcessEnvironmentDefault`
- `ChildProcessEnvironmentExceptions`

`MSIIs64Bit` has been added to the MSI-related settings. On a 64-bit operating system, ThinApp installs a 32-bit application in the `ProgramFiles(x86)` directory. Enable `MSIIs64Bit` if you want to install a 64-bit MSI package in the `ProgramFiles` directory.

For more information, see the ThinApp Package.ini Parameters Reference Guide.

SDK Changes

Enhancements to the SDK include the following.

Support for 64-bit SDK – The ThinApp SDK now includes a separate 64-bit DLL file (`ThinAppSDK64.dll`), thereby eliminating the need for `ThinAppSDKSrv.exe` on 64-bit operating systems. On a 64-bit OS, the users can register the 64-bit SDK DLL (`regsvr32 ThinAppSDK64.dll`) and use their scripts or applications to access ThinApp packages.

Registry key path changes – For packages built with ThinApp version 4.x or earlier, there are no changes if using registry APIs from the ThinApp SDK.

However, for 32-bit packages built with ThinApp 5.0, 32-bit keys are redirected to `HKLM\Software\Wow6432Node`.

For example:

- Native – `HKLM\Software\7-Zip`
- ThinApp – `HKLM\Software\Wow6432Node\7-Zip` (passed to the SDK as an argument)

There is no change to the registry path for a 64-bit application.

Virtual CMD

By default, ThinApp captures a 32-bit virtual command prompt. To start a 64-bit virtual command prompt, in the `Package.ini` file, change

```
[cmd.exe] Source=%SystemSystem%\cmd.exe
```

to

```
[cmd.exe] Source=%SystemSystem(x64)%\cmd.exe
```

Support for 64-Bit Applications

ThinApp 5.0 introduced support for 64-bit applications.

Note: If the ThinApp capture contains a 32-bit and a 64-bit executable, a file with the `.alt` extension is generated in the project's `bin` folder. Because the PDC is usually 32-bit and therefore unsuitable to start a 64-bit process, this ALT file provides an alternate-architecture PDC image to use as an entry point.

ThinApp Practical Exercises

These exercises take you through various ThinApp use cases, such as preparing a capture machine, packaging an application, and integrating ThinApp with other products and components in the Horizon 7 editions.

Note: To obtain a copy of ThinApp for these exercises, see Obtaining ThinApp.

ThinApp Packaging Process

Setup Capture begins with a prescan of the operating system to provide a baseline. When the prescan is complete, you install and configure the application and take a postscan. The prescan and postscan use OS-level snapshots that record the registry, file and folder structure, and other components affected by the application installation. After the postscan, the Setup Capture wizard asks you to configure the entry points, PDC, inventory name, access control, sandbox location, isolation mode, and compression. The baseline from the prescan is compared to the postscan, and all differences are recorded into the ThinApp project. The project also incorporates the information provided by you during the Setup Capture wizard.

Prepare the Capture Machine

You need a *clean capture machine* on which to install ThinApp and capture your applications. A clean capture machine includes only a basic installation of the Windows operating system, as well as the most recent Service Pack. Select the oldest common version of Windows that your users are using to ensure that the application captures include all required files during installation.

Capturing from a clean machine ensures that all the files and components necessary for the application are detected by Setup Capture. If an application installer looks for a certain version of a DLL file and the capture machine finds it in the local operating system, that DLL is not included in the virtualized application package.

Install the latest version of ThinApp onto this clean capture machine.

VMware Workstation™ is a recommended platform to capture and build ThinApp virtual applications. You can take a snapshot of a Workstation VM in its clean state, and revert to this snapshot whenever you capture a new application. The evaluation version of ThinApp includes a Workstation license.

Exercise 1: Package an Application Using Setup Capture

After you have prepared a clean capture machine and installed the latest version of ThinApp, you are ready to package an application using Setup Capture.

1. Select **Start** > **Programs** > **VMware** > **ThinApp Setup Capture**.

   Watch the Quick Start video, and use the contextual help for detailed guidance throughout the process.

   Click **Next**.

FIGURE 7: ThinApp Welcome Page

2. Click **Prescan** to create a baseline of the system environment.

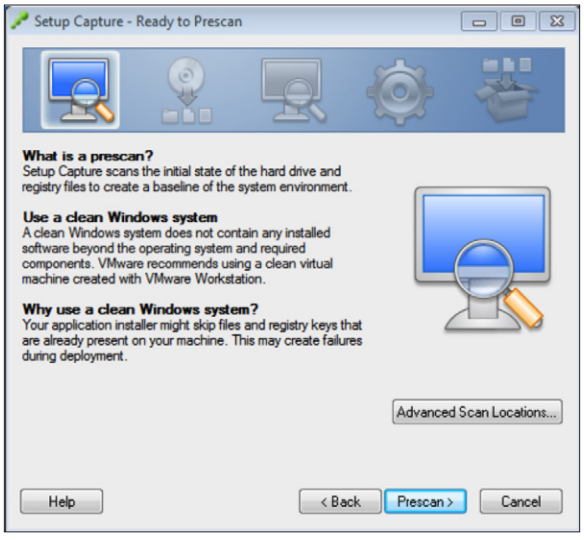

FIGURE 8: ThinApp Ready to Prescan Page

3. At the Install Application page, copy **`notepad.exe`** from **`C:\Windows`** to the desktop and click **Postscan**.

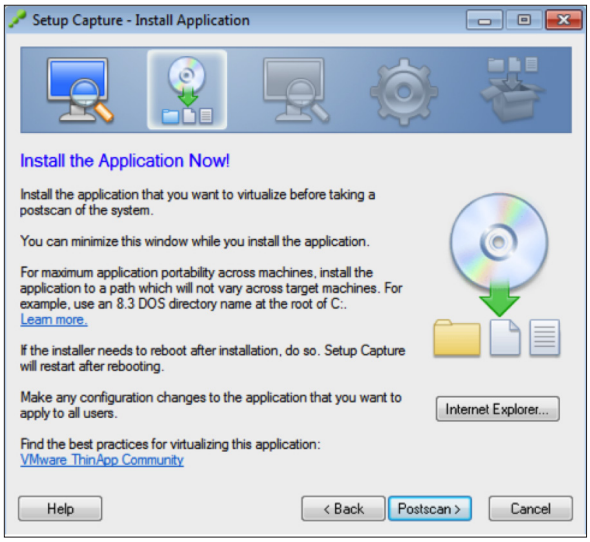

FIGURE 9: ThinApp Install Application Page

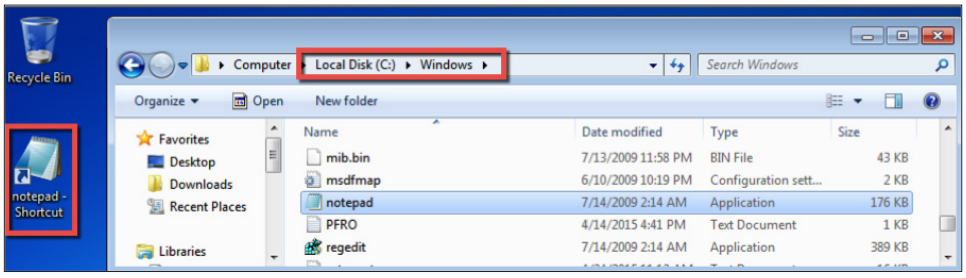

FIGURE 10: Copying Notepad.exe to Desktop

You will see the following page after you click **Postscan**.

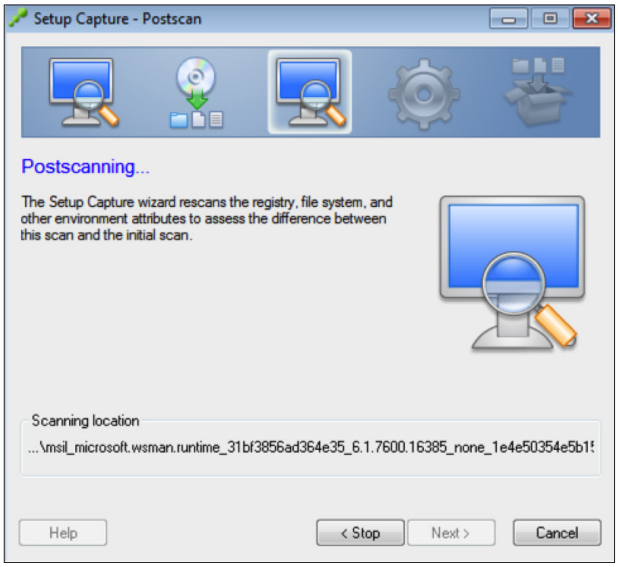

FIGURE 11: ThinApp Postscan Page

4. Select the entry point you want to use for your virtual application and click **Next.**
   Note: In this lab, there is only one entry point available for Notepad and it is preselected.

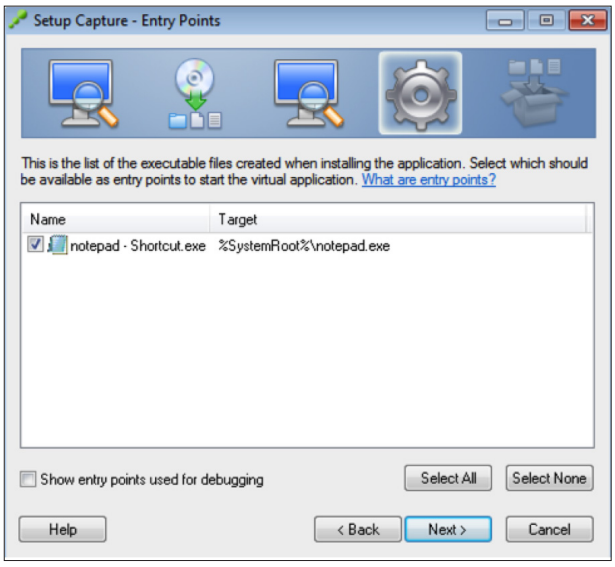

FIGURE 12: ThinApp Entry Points Page

5. You will see the Manage with VMware Workspace page. Keep the default selection and click **Next**.

   Notes

   - The wizard refers to VMware Workspace, however, this applies to VMware Identity Manager (formerly known as VMware Workspace).
   - For more information about ThinApp and VMware Identity Manager, see Enable a ThinApp Package for Use in VMware Identity Manager.

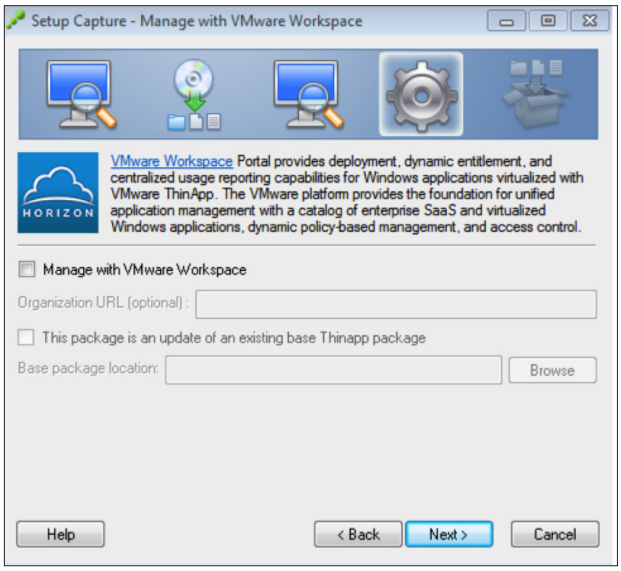

FIGURE 13: ThinApp Manage with VMware Workspace Page

6. Keep the default selection and click **Next**.

   Note: For more information about ThinApp and groups, see Use Active Directory Groups to Authorize ThinApp Packages.

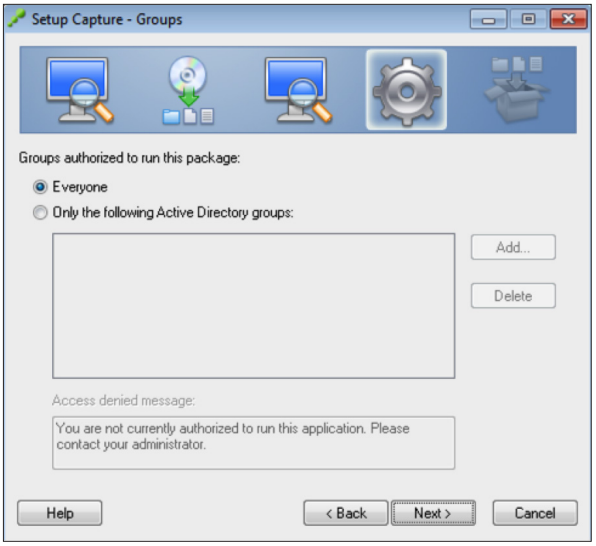

FIGURE 14: ThinApp Groups Page

7. Keep the default selection and click **Next**.

   Note: For details on the Setup Capture Isolation page, see Specify the Isolation Mode.

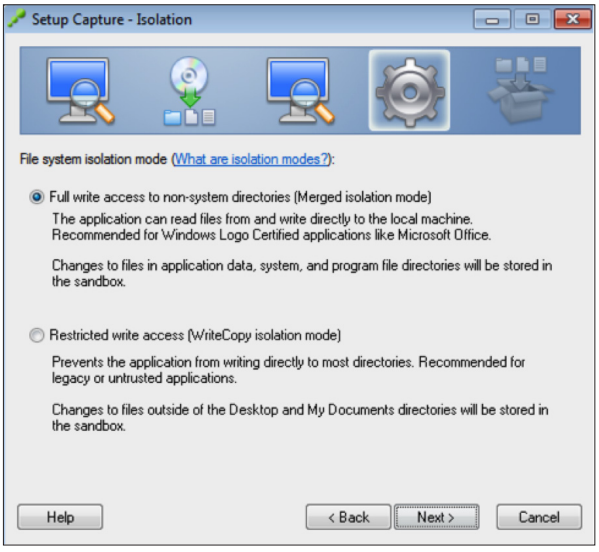

FIGURE 15: ThinApp Setup Capture Isolation Page

8. Keep the default selection and click **Next**.

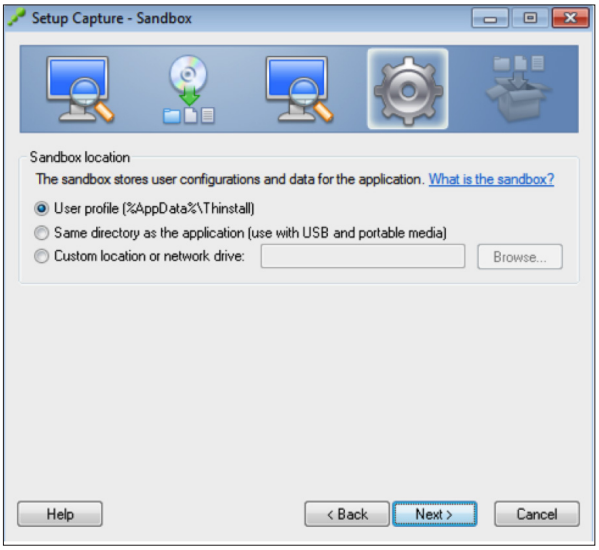

FIGURE 16: ThinApp Sandbox Page

9. Select an option on the Quality Assurance Statistics page and click **Next**.

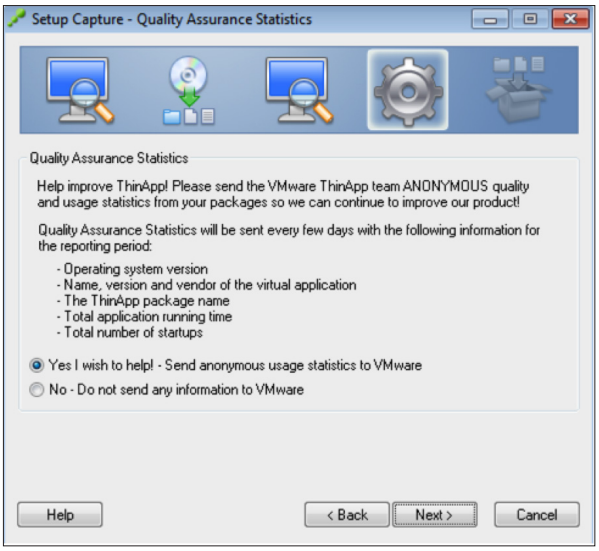

FIGURE 17: ThinApp Quality Assurance Statistics Page

10. Enter an inventory name for your virtual application and keep the default project location. Click **Next**.

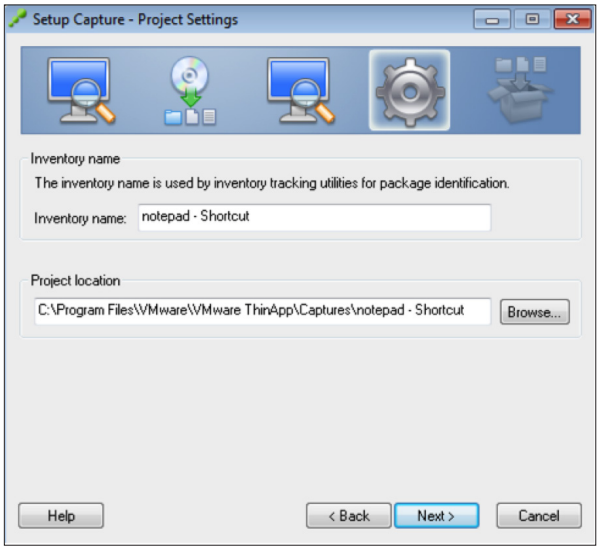

FIGURE 18: ThinApp Project Settings Page

11. Confirm details on the Package Settings page and click **Save**.

FIGURE 19: ThinApp Package Settings Page

Setup Capture saves the project files to the chosen directory.

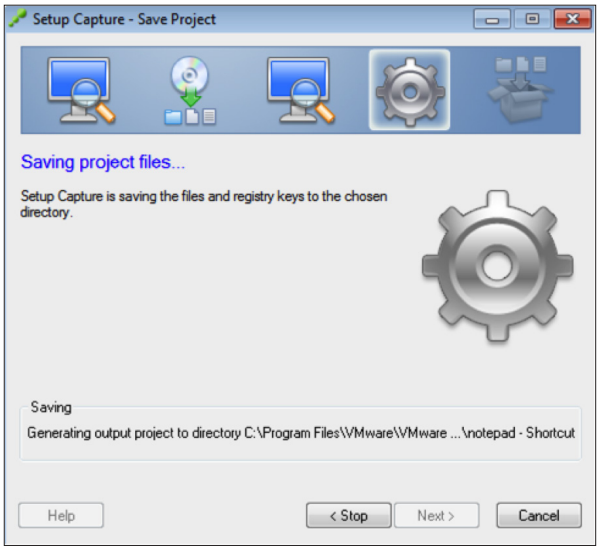

FIGURE 20: ThinApp Setup Capture Save Project Page

12. Click **Build** to create the virtual package.

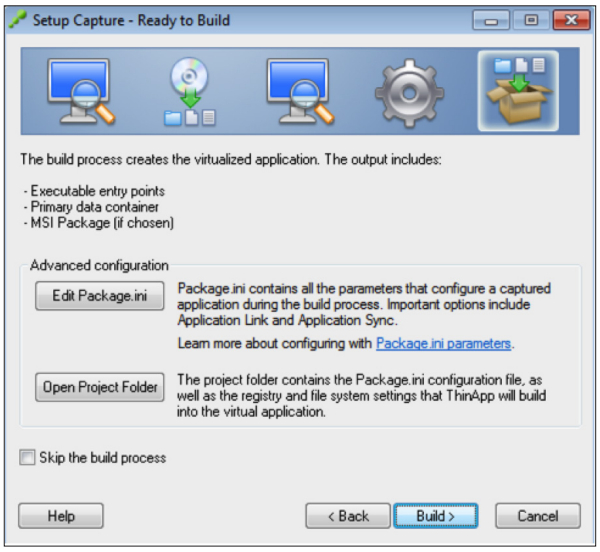

FIGURE 21: ThinApp Ready to Build Page

13. After the build process completes, click **Finish** to close the wizard.

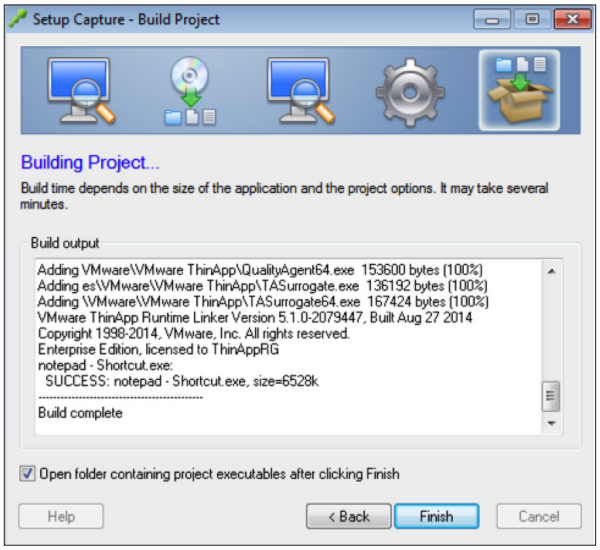

FIGURE 22: ThinApp Build Project Page

The project folder containing the executable opens by default after you click Finish.

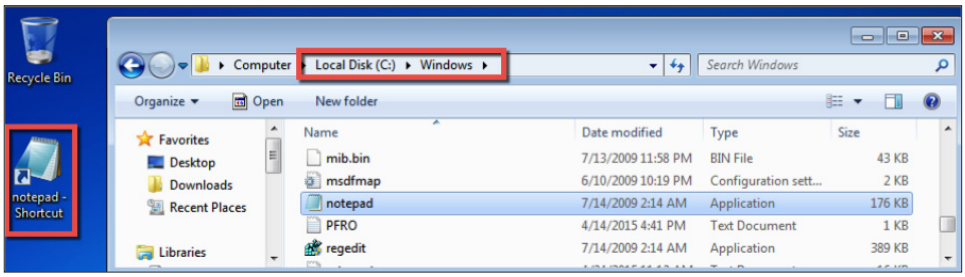

FIGURE 23: Virtual Application in Project Bin Folder

Special Considerations

The following sections provide more detail on certain features when advancing through the Setup Capture wizard.

Specify the Isolation Mode

Isolation modes allow you to control which host machine resources are visible to a virtual application. In the Setup Capture wizard, you can select Merged or WriteCopy isolation modes. The default directory isolation mode selection is **Full write access to non-system directories (Merged isolation mode)**. For more information about isolation modes, see ThinApp Architecture and Isolation modes.

Select the Primary Data Container

The PDC holds the virtual environment and the ThinApp runtime. In the Package Settings page, you can select an entry point or a separate DAT file as the PDC. If you select a separate data container, you must store the DAT file and the entry points in the same folder. If the project is larger than 2 GB, you must use a DAT file because of the Windows limitation for EXE files. Windows does not execute EXE files larger than 2 GB.

Exercise 2: Use Active Directory Groups to Authorize ThinApp Packages

The process of deploying virtualized applications offers administrators control and flexibility over which machines and users receive them. Using Active Directory allows an organization to use the existing processes and controls for group-based security. In addition to these organizational controls, ThinApp allows an administrator to embed access control into the package. This access control mechanism is hidden from the end user when the package is built, so it is impossible to identify or remove before the application is started. Access control travels with the package if it is moved between endpoints after deployment.

Capture an application and proceed through the Setup Capture wizard until you see the Groups page.

1. When the Groups page appears, select **Only the following Active Directory groups** and click **Add**.

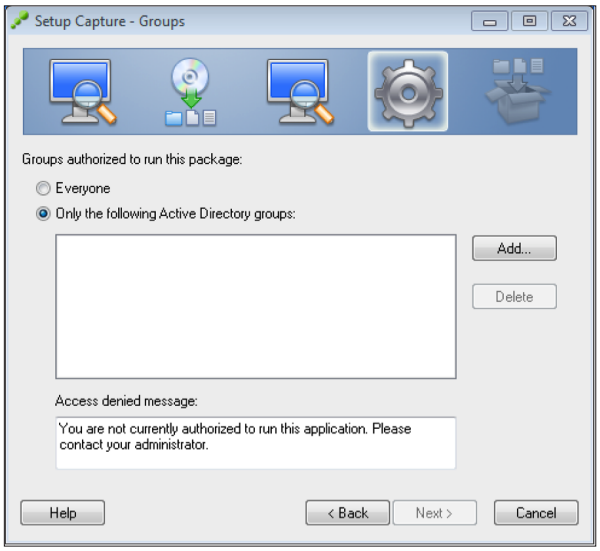

FIGURE 24: Using Active Directory to Manage Group Access

The Select Groups page opens.

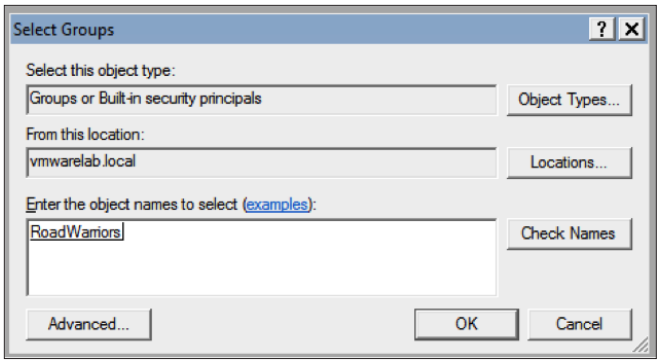

FIGURE 25: Selecting Which Groups and Object Types Have Access

2. Select the Active Directory groups that you want to authorize for access to the application in the following way.
   - To specify Active Directory objects, click **Object Types**.
   - To specify a location in the Active Directory forest, click **Locations**.
   - To search object names, enter a name in the Enter the Object Names to Select text box and click **Check Names**. (You can click the **examples** link in the dialog box for an example.)
   - For more advanced Active Directory queries, click **Advanced** and use the **Common Queries** tab to search for groups according to names, descriptions, disabled accounts, passwords, and days since the last login.

3. Click **OK**.

   The following example shows that only the RoadWarriors Active Directory group is able to start this application.

   Note: You can also customize the access denied message that appears if a user without access attempts to start the application.

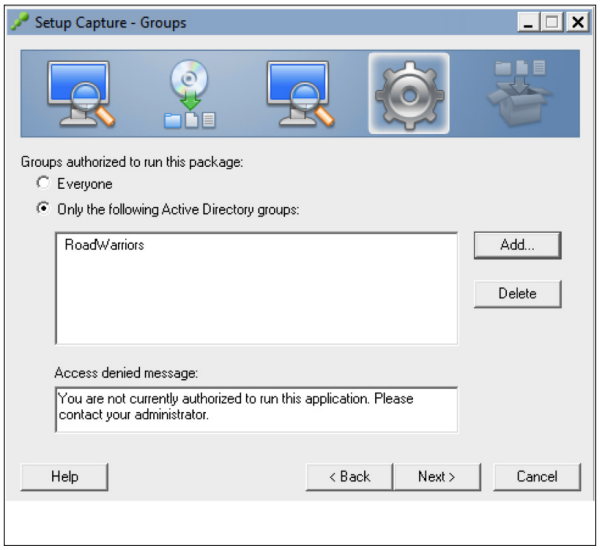

FIGURE 26: Restricting Access to an Active Directory Group

After you have configured the authorized Active Directory groups, click **Next** to proceed through the Setup Capture wizard.

Enable a ThinApp Package for Use in VMware Identity Manager

In ThinApp 5.1, the Setup Capture wizard includes an option to enable VMware Identity Manager to manage a ThinApp package. If you select this option, when a user starts the ThinApp package, ThinApp checks that the VMware Identity Manager Desktop application is present and contacts a VMware Identity Manager server. If successful, the application authorizes access to the ThinApp package through entitlements set up by the IT administrator in VMware Identity Manager.

Note: The wizard refers to VMware Workspace, however, this applies to VMware Identity Manager (formerly known as VMware Workspace).

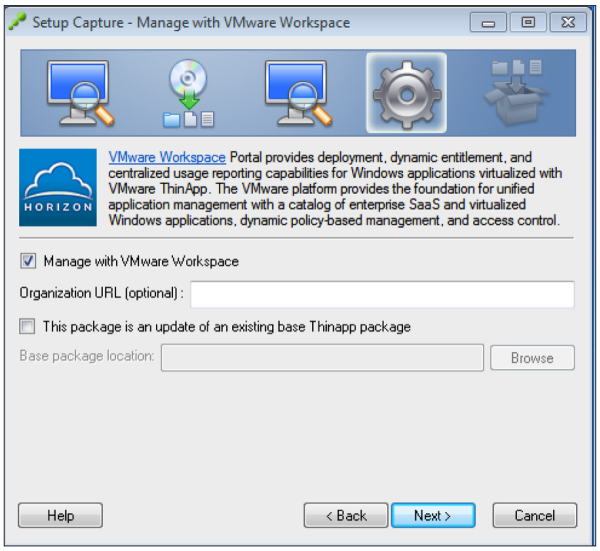

FIGURE 27: Selecting VMware Identity Manager to Manage ThinApp Packages

The VMware Identity Manager for Desktops application must be installed on the user's desktop before the user can start a ThinApp package that is enabled for VMware Identity Manager. For more information, see *Installing Identity Manager Desktop* in Using VMware Identity Manager Desktop.

The **Organization URL** field in the Manage with VMware Workspace page allows you to supply your organization's VMware Identity Manager URL. If the VMware Identity Manager for Desktops application is not installed on the desktop when the user tries to start the ThinApp package, this URL leads to the VMware Identity Manager, which initiates its installation. Entering a value in this field facilitates the automation of the VMware Identity Manager for Desktops installation.

Note: If you enable the ThinApp package for VMware Identity Manager, the Setup Capture Groups wizard page is skipped, because VMware Identity Manager manages entitlement of users to the ThinApp package. In VMware Identity Manager, you can set up package entitlement of Active Directory users or groups or of a Horizon 7 group that you create.

For more information on how to use VMware Identity Manager to manage ThinApp packages, see Use ThinApp Packages with VMware Identity Manager.

Enable ThinApp Packages Created Prior to ThinApp 4.7 for VMware Identity Manager

If you have a ThinApp project created with a version of ThinApp earlier than 4.7 and you want to use it with VMware Identity Manager, you can install the latest version of ThinApp, add the VMware Identity Manager parameters to the project's `Package.ini` file, and rebuild the package.

The `Package.ini` file has three parameters for enabling VMware Identity Manager.

Note: The `Package.ini` file refers to VMware Workspace, however, this applies to VMware Identity Manager.

```
;-------- VMware Workspace Portal Parameters ----------
AppID=genid
NotificationDLLs=HorizonPlugin.dll
HorizonOrgUrl=https:\\customerdomain.horizonmanager.com
```

FIGURE 28: Package.ini Parameters for VMware Identity Manager Enablement

- **AppID=genid** – Assigns a random GUID to the ThinApp package. VMware Identity Manager uses the GUID to identify the package in the ThinApp repository and to check entitlement. The GUID is a unique identifier that can be used across all VMware Horizon programs and services touching the ThinApp package.
- **NotificationDLLs** – Specifies the DLL file (`HorizonPlugin.dll`) that the ThinApp runtime calls to check with the Horizon Agent for entitlement to run the application.
- **HorizonOrgUrl** – Your organization's VMware Identity Manager URL, which you have the option of specifying in Setup Capture. In this example, a generic URL is entered (`https://customerdomain.horizonmanager.com`). The Horizon 7 service at the URL initiates a VMware Identity Manager for Desktops application installation if it is not already installed when the user tries to start the ThinApp package.

For more information, see the VMware knowledge base article Enabling for Horizon a previously packaged VMware ThinApp virtual application, without relinking (2030595).

Instead of rebuilding the project after enabling the VMware Identity Manager parameters, you can use the `relink -h` command from the latest version of ThinApp. This will update the package to the latest ThinApp runtime and also enable VMware Identity Manager.

For more information, see the VMware knowledge base article VMware ThinApp relink command to update the ThinApp runtime of a virtual application (2030593).

Exercise 3: Modify the Package.ini File Settings

The last step of the Setup Capture wizard prompts you for advanced configuration before building the package. The `Package.ini` file contains configuration settings and resides in the captured application's project folder.

Capture an application and proceed through the Setup Capture wizard until you see the Ready to Build page.

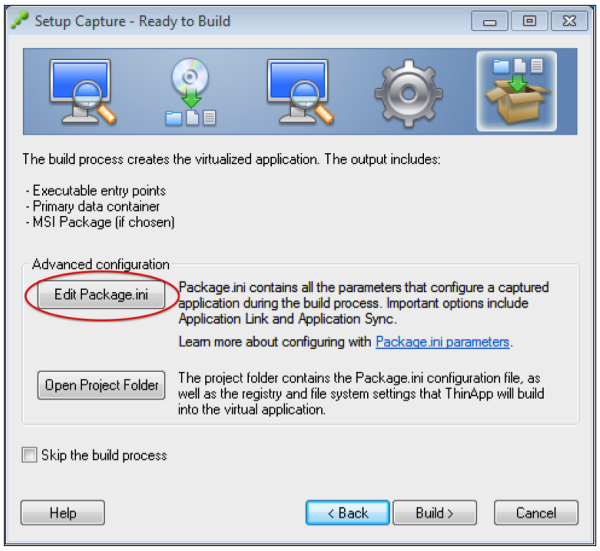

FIGURE 29: Editing the Package.ini File

Perform the following steps to edit the `Package.ini` file:

1. Click **Edit Package.ini**.
2. To modify a parameter, remove the semicolon at the beginning of the line to enable an option or edit existing settings.

   For example, to activate the `RemoveSandboxOnExit` parameter, delete the semicolon at the beginning of the line so that `RemoveSandboxOnExit=1` is enabled in `Package.ini`.
3. Save the file.
4. In the Setup Capture wizard, click **Build** to complete the process and generate the ThinApp package.

   Subsequent sections refer to different `Package.ini` modifications. When making changes to these parameters, you edit the `Package.ini` file and rebuild the package to embed the new settings within the package.

Important: You do not need to use the Setup Capture wizard to edit a `Package.ini` file or rebuild a package. You can edit the file directly with Windows Explorer, and you can double-click `build.bat` in the same project folder once you have made required changes.

Exercise 4: Capture IE 6 and Use ThinDirect with Setup Capture

Since ThinApp 4.6, you can capture IE 6 on Windows XP. With ThinApp 5.1, you can virtualize many versions of Internet Explorer, up to IE 11. Virtualizing IE isolates the Web browser and provides support for legacy applications. You can run virtualized IE and native IE versions concurrently on the same desktop.

This section describes how to capture IE 6 from Windows XP. Setup Capture copies a defined set of files and registry settings into the project without needing to actually install IE 6.

Note: Microsoft no longer supports IE 6, but ThinApp does support it as a legacy application so long as it is captured in this way.

For more information, see the VMware knowledge base article Virtualizing Internet Explorer 6 with ThinApp 4.6 and later (1026565).

Important: You must use a Windows XP capture machine to virtualize IE 6.

1. Run **ThinApp Setup Capture** and click **Prescan**.
2. In Setup Capture, click the **Internet Explorer** button on the Install Application page.

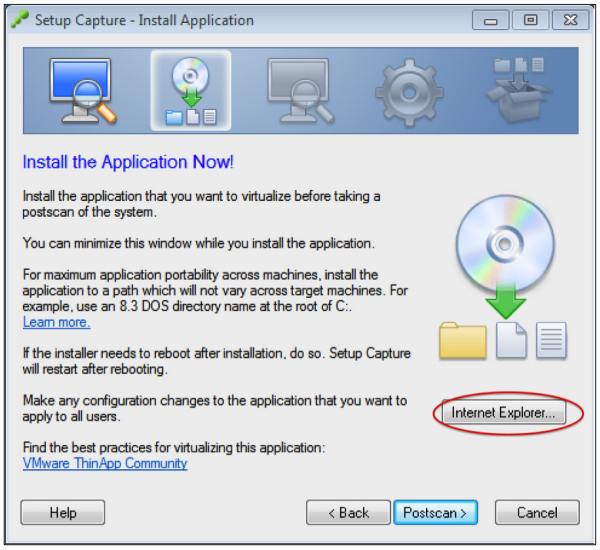

FIGURE 30: Installing Internet Explorer Using Setup Capture

3. Select **Include an entry point for a fully virtualized Internet Explorer** and click **OK**.

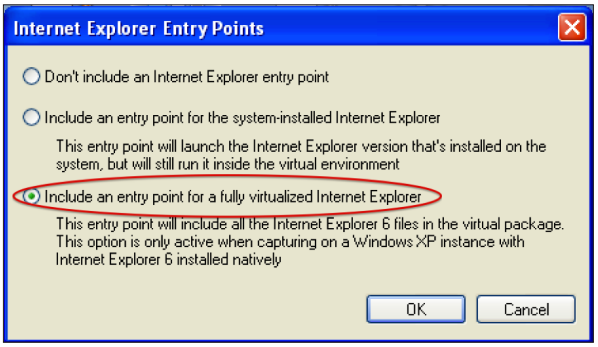

FIGURE 31: Capturing Internet Explorer 6

4. Add browser plug-ins or modifications and click **Postscan**. During the postscan, ThinApp copies the required IE 6 files and settings from the Windows XP capture machine into the project folder.
5. Follow the prompts until the Native Browser Redirection page is displayed, with a text box in which you can enter the Web sites and pages to redirect to this browser. These entries are added to the `ThinDirect.txt` file and can be edited manually. You can also add entries dynamically to the end users' endpoints using Active Directory GPOs. For more information, see the VMware blog post Virtualizing Internet Explorer, What Works?

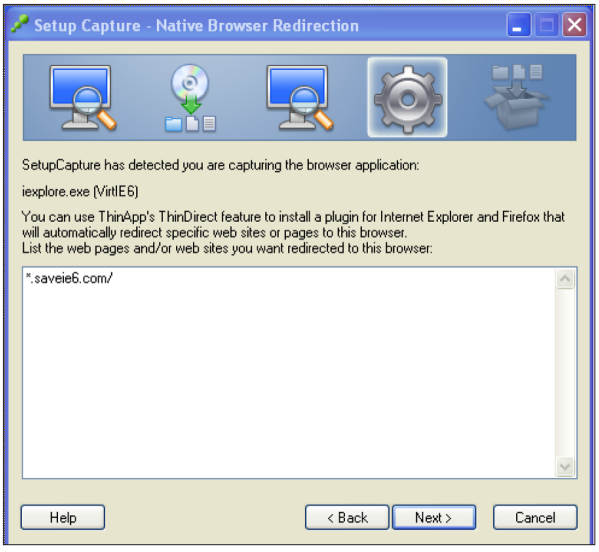

FIGURE 32: Specifying Sites and Pages for Redirection

6. Follow the prompts until you can click **Build** to create the ThinApp package.

This procedure results in an IE 6 package, either EXE- or MSI-based, that can be deployed to Windows Vista or later and run in parallel with other virtualized or native browsers. The ThinDirect feature gives administrators the flexibility to seamlessly redirect end users to particular browsers based on a whitelisting model.

For details on `ThinDirect.txt` syntax, see the VMware knowledge base article ThinDirect.txt syntax for redirecting Web sites from native Internet Explorer to a virtual browser such as virtual Internet Explorer 6 (1026635).

Exercise 5: Set Isolation Modes

Isolation modes can control the host file machine resources that are visible to a virtual application. ThinApp has three isolation modes: Full, WriteCopy, and Merged. For definitions, see *Isolation modes* in ThinApp Components and Terminology.

The following steps show how to set different isolation modes in a captured application.

1. Create a new ThinApp package using Notepad++. You can download the Notepad++ application from https://notepad-plus-plus.org/. Proceed through Setup Capture until you reach the **Ready to Build** page.
2. At the **Ready to Build** page, click **Open Project Folder**.

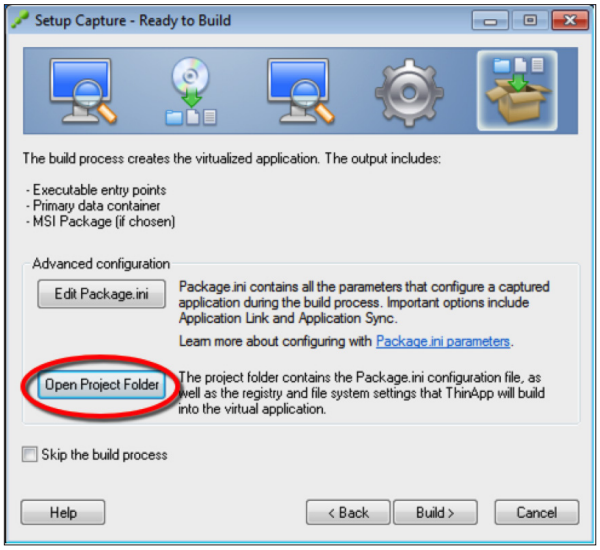

FIGURE 33: Opening a Project Folder in Setup Capture

3. Edit the `##Attributes.ini` file in the `%Desktop%` directory to set `WriteCopy` isolation.

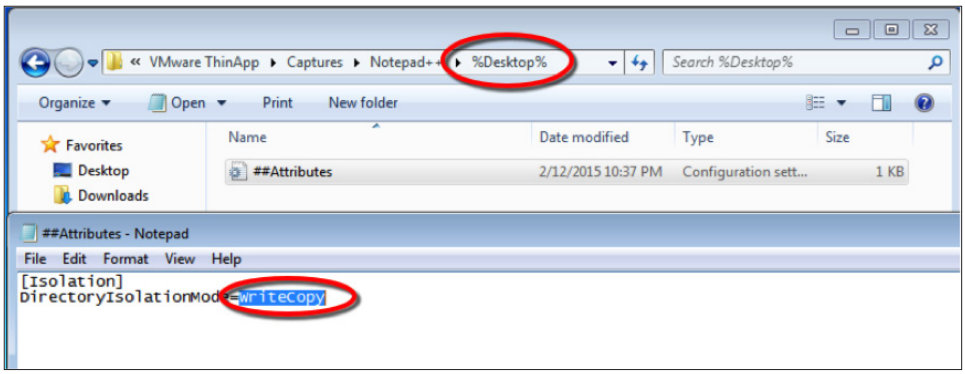

FIGURE 34: Setting the Isolation Mode to WriteCopy

4. Select **Build** to build the project.
5. Run virtual Notepad++. You can run it on the capture machine.
6. Create a document, and save it to the desktop. Is the document visible in virtual Notepad++? Can you see it on the desktop?

   The file is available in virtual Notepad++, but you cannot view it on the host desktop. WriteCopy mode allows virtual Notepad++ to view host data, but any changes made are redirected to the sandbox.

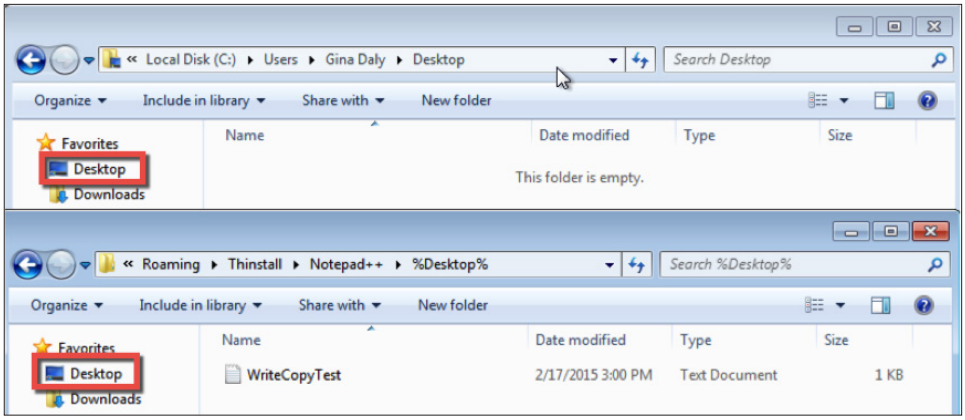

FIGURE 35: Comparison of Native Desktop Folder and Virtual Desktop Folder

7. Change the isolation mode to **Merged**, and rebuild the package.
8. Delete the sandbox located in `%APPDATA%\Thinstall`.
9. Repeat Steps 3 to 6.

   Open the document using the native Notepad++ application. Do you see the content you just created?

   The file is available to the native version of Notepad++, because Merged mode allows data to be stored on the host desktop.
10. Change the isolation mode back to WriteCopy, and rebuild the project.

11. Start virtual Notepad++.
12. Try to open the existing document. What is the result?

    The file is available to virtual Notepad++ when using WriteCopy isolation mode.
13. Change the content of the document and save it.
14. Open the document using the native version of Notepad++. Do you see the changed text? Check it again with virtual Notepad++.

    You cannot see the updated text when using native Notepad++, because the changes made have been redirected to the sandbox. However, the changes are available in virtual Notepad++.
15. Change the isolation mode to **Full**, and rebuild the package.
16. Delete the sandbox.
17. Repeat Steps 12 and 13.
18. Create a new document with the same name and with different content.

    Did the existing document text change? The existing document text does not change even though we have created a file with the same name. This new file is redirected to the sandbox, because Full isolation mode does not allow the virtual application to read or modify elements on the host machine.

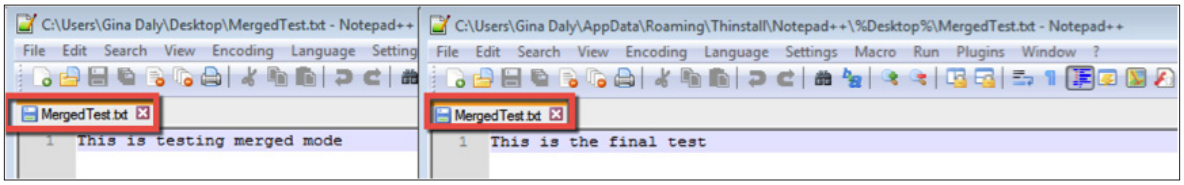

FIGURE 36: Comparing Text Files in Native and Virtual Notepad

In this exercise, you practiced setting different isolation modes for a ThinApp package and learned how these isolation modes affect the behavior of that application.

Exercise 6: Deploy ThinApp Packages

Deploying ThinApp packages is simple, because you do not actually install the application, and interaction with the local operating system is minimal. Deployment involves selecting an application delivery model and integrating the virtualized applications into the desktop for end-user accessibility. For information about selecting the type of deployment and delivery mode and registering applications, see Appendix B.

Since View 4.5, you can deploy and register ThinApp packages from within the View Administrator console. The steps to deploy ThinApp packages with View environments are covered in Use ThinApp Packages with View.

When the build process has completed, the ThinApp package is placed in the `bin` folder of the project directory. To deploy the package, you copy the contents of the `bin` folder to an appropriate location. Portability of these packages gives administrators and end users significant flexibility for deployment. To secure these packages with Active Directory groups, see Use Active Directory Groups to Authorize ThinApp Packages.

The following scenarios demonstrate the portability of ThinApp packages and the two primary modes of deployment.

For these exercises, you can create a new package or you can use the Notepad++ package created in the previous exercise.

Deploy a ThinApp Package Locally

1. Copy the EXE- or MSI-based ThinApp package to a local file system or removable media.

   Note: Double-clicking an MSI-based ThinApp package will install it and create desktop or Start menu shortcuts.
2. Start the application by double-clicking its EXE file or its MSI-created shortcut icon, and test its function.

Deploy a ThinApp Package Remotely

1. Copy an EXE-based ThinApp package to a remote file share.

   Note: It is also possible to deploy an MSI-based ThinApp package remotely. For more information, see Create MSI Packages for Local and Remote Deployment.
2. Create a shortcut from the package to your desktop.
3. Start the application, and test its function.

Deploy ThinApp Packages on Multiple Operating Systems

1. Copy the EXE package to a Windows 7 machine.
2. Start the application and test its function.
3. Copy the EXE package to a Windows Server 2008 machine that has been configured as a Microsoft Remote Desktop Services Host (RDSH).
4. Publish the application as a remote app using the RDSH server.
5. Start the application and test its function.
6. Copy the EXE package to a Citrix XenApp application server.
7. Publish the application from the Citrix XenApp application server.
8. Start the application and test its function.

Deploy MSI Packages Using Active Directory

For this exercise, you will need an MSI-based ThinApp package. You can enable the `MSIFilename` parameter in `Package.ini` from the existing Notepad++ package. The MSI package is created after you rebuild the project.

See the following blog posts regarding GPO deployment:

- Notes and Considerations on Deploying ThinApp Packaged Applications via Active Directory Group Policies
- How to use Group Policy to remotely install software in Windows Server 2008

1. Create a GPO to assign an application.
2. Configure the security groups you want this policy to apply to.
3. Select the MSI-based ThinApp package to deploy using the newly created GPO.
4. Publish the ThinApp package to make it available for installation from the Programs and Features applet in Control Panel.
5. Log in to a machine or as a user specified by the GPO and confirm installation by looking in the Programs and Features applet of the Control Panel.
6. Start the application and test its function.

Exercise 7: Use ThinApp Packages with View

The View Administrator assigns ThinApp packages to individual desktops or pools of desktops to allow for streamlined application deployment. The requirements to use this method are

- MSI-based ThinApp packages
- A file share
- View 4.5 or later

To review the setup and operation of ThinApp assignments in View, refer to the How to Add a ThinApp Application to a View environment video.

Create MSI Packages for Local and Remote Deployment

ThinApp packages used for assignment through View Administrator must be in MSI format. The default MSI selection in Setup Capture creates a package that can be deployed only in local deployment mode through View Administrator. To deploy the package in remote deployment mode (from a file share), the `MSIStreaming` parameter must be set to `1`.

1. To modify an existing package for remote deployment, edit the `Package.ini` file to enable `MSIStreaming=1` and rebuild the package.

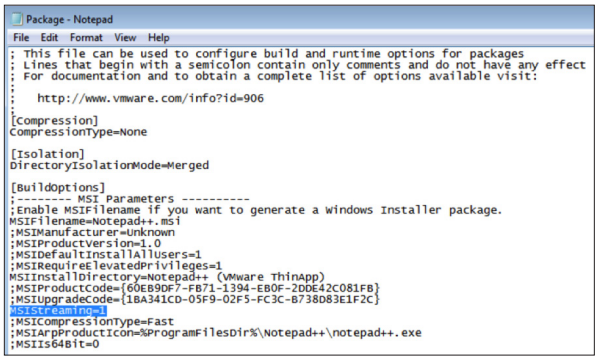

FIGURE 37: MSI Parameters in Package.ini

2. Copy all the files from the `bin` directory to the file share that will serve as the View ThinApp repository.
3. Set permissions on the file share so that View Administrator and end users can access it. For information on security recommendations, see the Horizon 7 documentation.

Create a View ThinApp Repository and Populate It with ThinApp Packages

The next step is to create a View ThinApp repository and place the previously created MSI packages in the repository, ready for assignment.

1. In View Administrator, navigate to **View Configuration** > **ThinApp Configuration** and click **Add Application Repository**.
2. Specify the display name, the file share path, and optionally provide a description. Click **Save**.

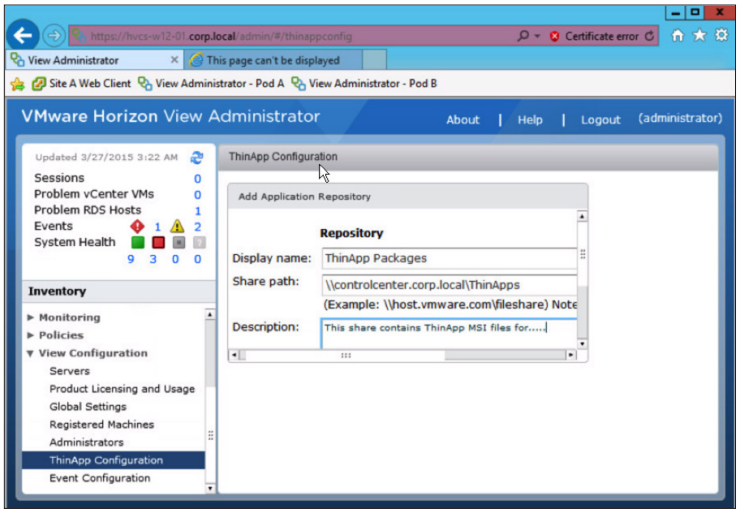

FIGURE 38: ThinApp Configuration in View Administrator

3. Navigate to **Catalog** > **ThinApps** and click **Scan New ThinApps**.

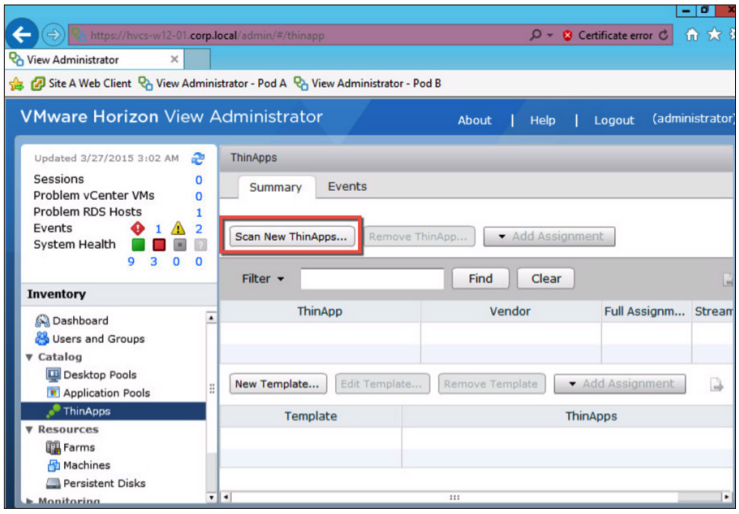

FIGURE 39: Scanning for New ThinApp Packages in View Administrator

4. Select the ThinApp repository that you just created.

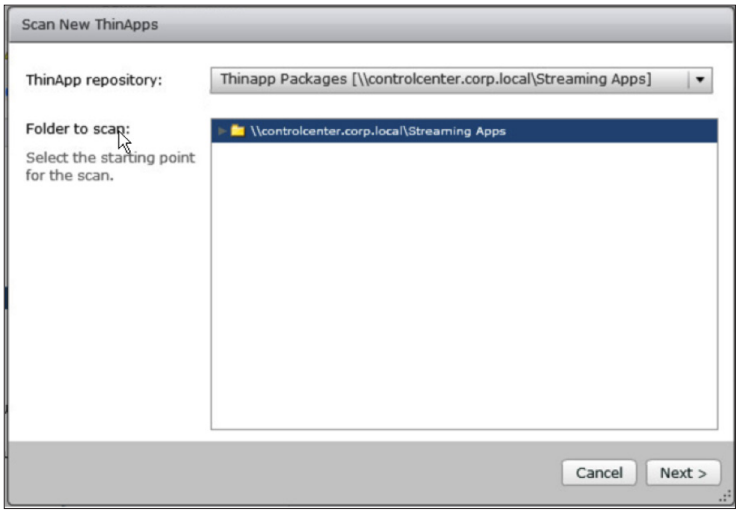

FIGURE 40: Select the Repository to Scan

5. Select the MSI packages to add to the View ThinApp repository, and click **Scan**.

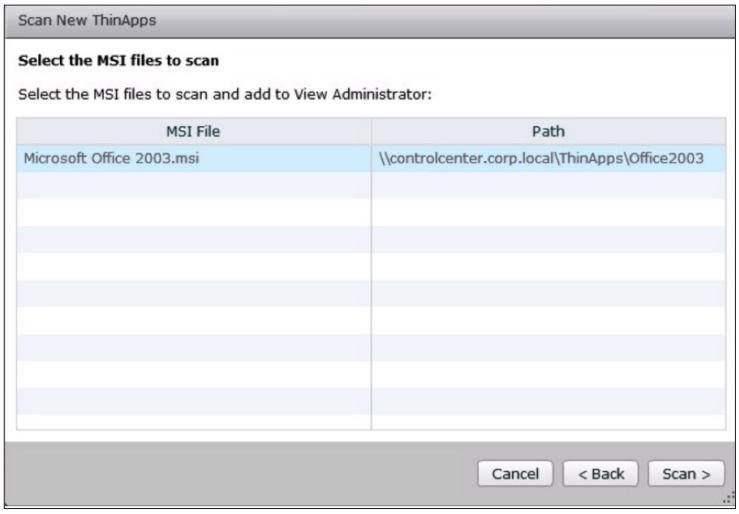

Figure 41: Select the MSI Files to Scan

Navigate to **Inventory** > **ThinApps** and click the **Summary** tab to see the newly scanned ThinApp packages.

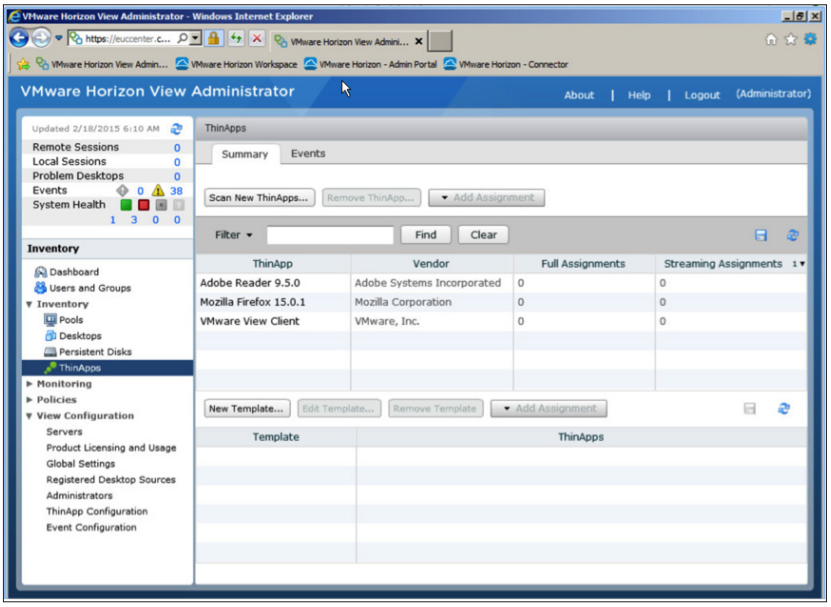

FIGURE 42: Scanned MSI Packages in View Administrator

Create ThinApp Assignments in View Administrator

After you have populated the ThinApp repository with MSI packages, you can assign those packages to desktops or pools.

1. (Optional) Create ThinApp templates, which are groups of ThinApp packages. A template allows you to make one assignment to a desktop or pool and deliver a group of applications instead of making multiple assignments. Click **New Template**, give it a name, and add specific ThinApp packages to populate the template.

2. Select either a ThinApp template or a single package and click **Add Assignment**.

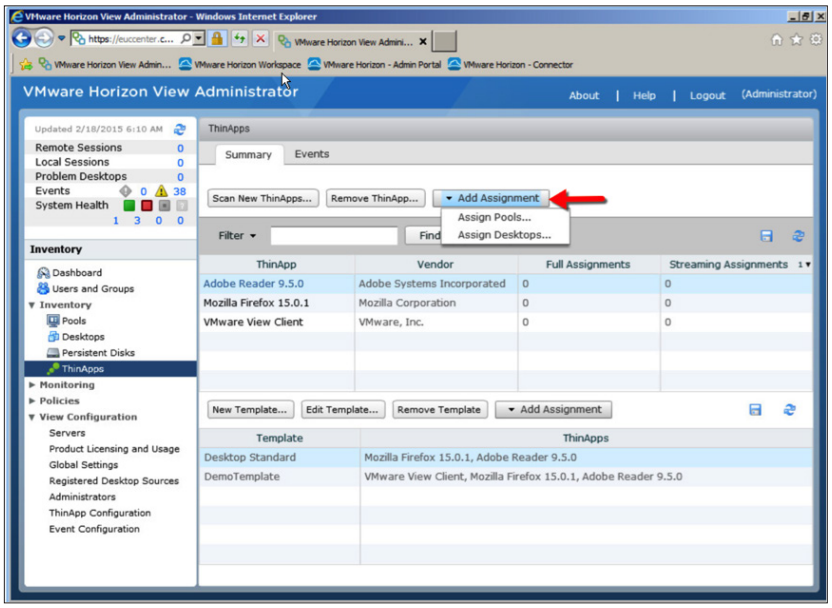

FIGURE 43: Adding an Assignment to a ThinApp Package

3. Select single or multiple desktops or pools. Click **OK**.

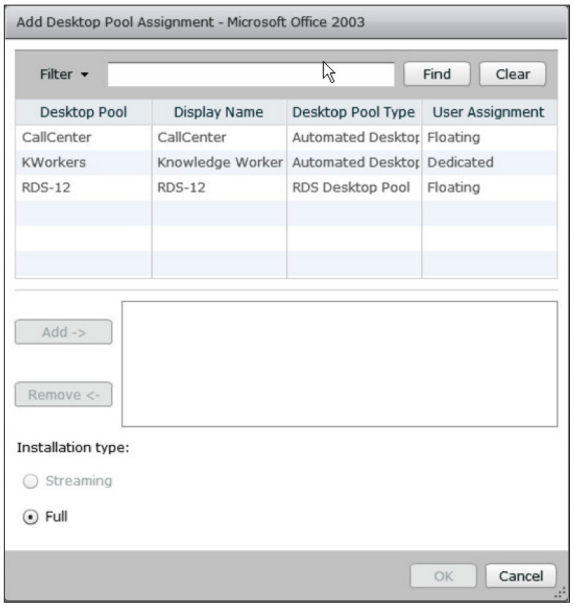

FIGURE 44: Assigning Desktop Pools in View

4. Navigate to **Monitoring** > **Events** to verify the assignment, and monitor progress by using the Events menu item.

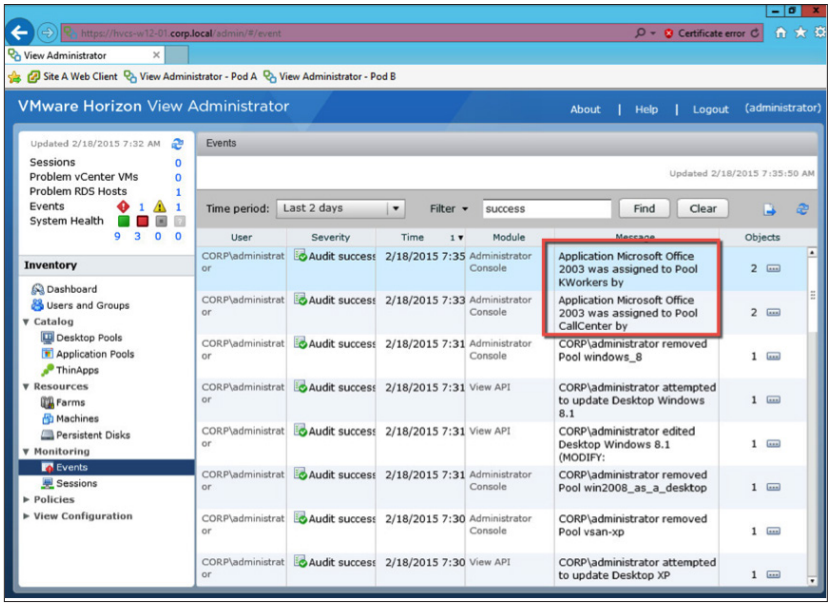

FIGURE 45: Events Applet in View

In this exercise, you reviewed how to set up a View ThinApp repository and make ThinApp assignments within View Administrator. Application registration happens automatically when ThinApp packages are deployed in this manner.

Use Script-Based Registration in View

To deploy ThinApp packages to View desktops without using pool or desktop assignments in View Administrator, you can use a Windows logon script and the **`thinreg.exe`** tool. You might need to do this if you want to assign a ThinApp package to a specific user, or if you are migrating from a non-View environment in which Windows logon scripts are already used for ThinApp package deployment.

For more information on using Windows logon scripts and Thinreg, see Script-Based Registration.

Use ThinApp Packages with App Volumes

App Volumes delivers applications almost instantly to end users and desktops, and supports the delivery of ThinApp applications. ThinApp applications address the issues of application conflicts and application-to-OS isolation, providing support for legacy applications and Web browsers. ThinApp packages can be delivered alongside native applications, bringing the benefits of both products to your environment. These benefits are discussed in App Volumes and ThinApp.

To integrate ThinApp with App Volumes, create an App Volumes AppStack from the App Volumes Manager and install the ThinApp packages on this AppStack during provisioning mode. After the AppStack has been created, assign or entitle the AppStack to users, groups, or computers in your VDI environment.

For more information, see the following VMware blog posts.

- VMware App Volumes and VMware ThinApp Combined: The Perfect Mix
- Using VMware App Volumes with ThinApp Packages

Exercise 8: Use ThinApp Packages with VMware Identity Manager

VMware Identity Manager provides end users with a single graphical interface from which to run multiple applications from different sources. This interface is accessed through single sign-on. Applications available to the user include SaaS applications, Windows applications, RDSH applications, Citrix-published applications, and ThinApp packages. In addition to applications, VMware Identity Manager provides end users access to View desktops.

VMware Identity Manager offers increased security for ThinApp packages, allowing administrators to restrict access to certain users as well as groups.

To deliver and manage ThinApp packages with VMware Identity Manager, you must have a ThinApp repository that contains the ThinApp packages, point to that repository, and sync the packages. After the synchronization, ThinApp packages are available in your VMware Identity Manager catalog to entitle to users and groups.

The exercises in this section include the following:

- Configure the ThinApp repository for VMware Identity Manager
- Entitle ThinApp packages to users
- Install the VMware Identity Manager Desktop application on client PCs
- Start ThinApp packages from VMware Identity Manager

The following exercises must be completed in the order presented.

For these exercises, you need

- VMware Identity Manager, installed and configured on a domain
- A file share
- ThinApp packages to place on the file share

For information on requirements and setting up and installing VMware Identity Manager in your environment, see the VMware Identity Manager documentation.

Configure the ThinApp Repository for VMware Identity Manager

The ThinApp repository (Windows Application Share) holds the ThinApp packages enabled for VMware Identity Manager. The Identity Manager Connector communicates ThinApp package metadata to VMware Identity Manager.

Before you set up entitlement to and management of the ThinApp packages in VMware Identity Manager, you must create the file share for the ThinApp packages and place the packages there.

For file share requirements, see *VMware Identity Manager Requirements for ThinApp Packages and the Network Share Repository* in the Setting Up Resources in VMware Identity Manager guide.

Each ThinApp package stored on the file share must have its own named folder to hold its files. Create a folder structure for the applications on the file share as follows:

`\\Server\sharename\virt_appname1`

`\\Server\sharename\virt_appname2`

The subfolder name for the virtual application does not have any particular restrictions, so you can customize it how you want.

For each ThinApp package you want in VMware Identity Manager, copy all EXE, ALT, and DAT files from the ThinApp project's `bin` directory to the named application subfolder in the ThinApp repository. VMware Identity Manager uses only EXE-based virtualized applications. You do not need to copy the MSI packages from the `bin` directories. However, for possible future use, you can also copy the MSI files. No errors will occur if you include the MSI packages in the ThinApp repository.

Note: Assign just read-only access to the file share so that applications can be downloaded to user desktops and application metadata can flow freely to the Identity Manager Connector, but users cannot change or delete the applications.

The following steps guide you through configuring a ThinApp repository for VMware Identity Manager.

1. Start a Web browser and navigate to `https://<IdentityManagerHostname>`.
2. Log in to the administrator console.

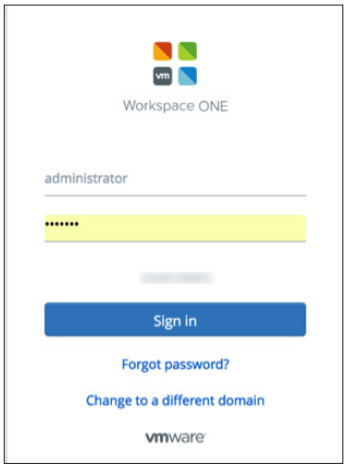

FIGURE 46: VMware Identity Manager Login

3. Select the **Catalog** tab.
4. Click **Manage Desktop Applications** and select **ThinApp Applications**.

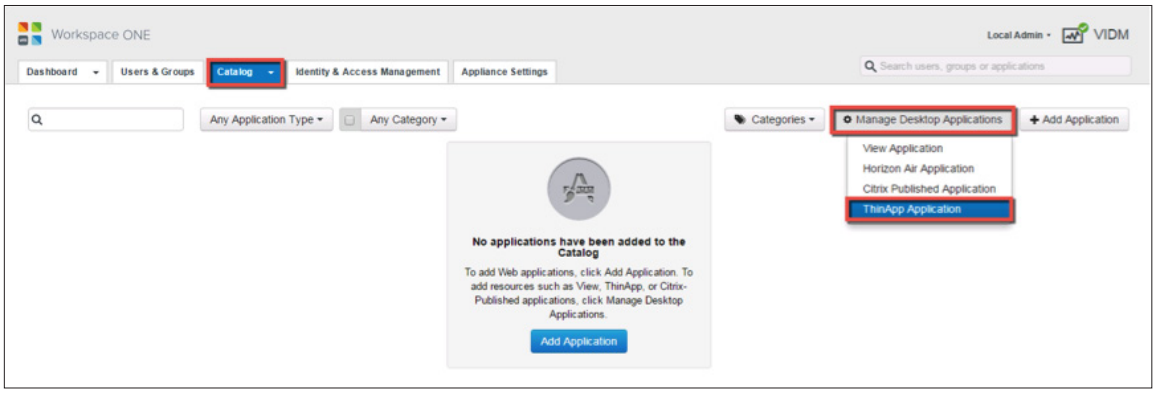

FIGURE 47: Adding ThinApp Applications

5. Select the **Enable packaged applications** check box and click **Save**.

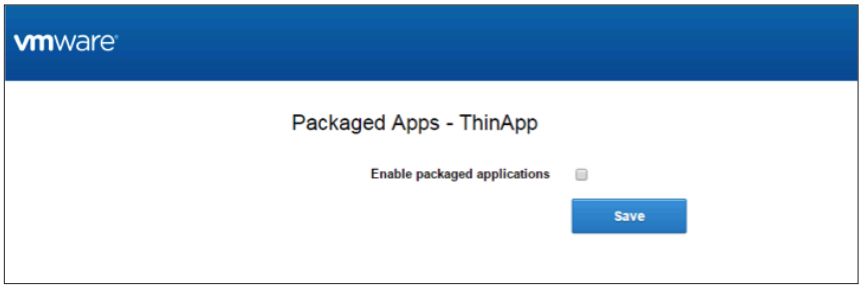

FIGURE 48: Enabling Packaged Applications

6. Provide the following details and click **Save**.
   a. Specify the path of the network file share where the packages are stored.
   b. Select how often VMware Identity Manager updates ThinApp packages by scanning the repository.
   c. If you are using a DFS share, select **Enable Account based access**. This option allows a DFS share to be used as the repository and allows non-domain members to download ThinApp packages using HTTPS. If you use account-based access, you can manage and deploy ThinApp packages on clients outside of your firewall.
   d. Specify a username for the share user. This user account must have read access to the network share.
   e. Specify the share password.

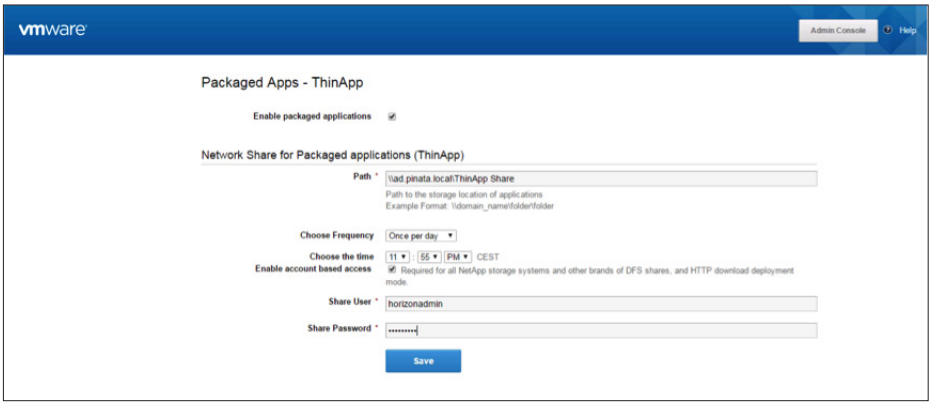

FIGURE 49: Configuring ThinApp Network Share

7. To synchronize the ThinApp repository to VMware Identity Manager, click **Sync Now**.

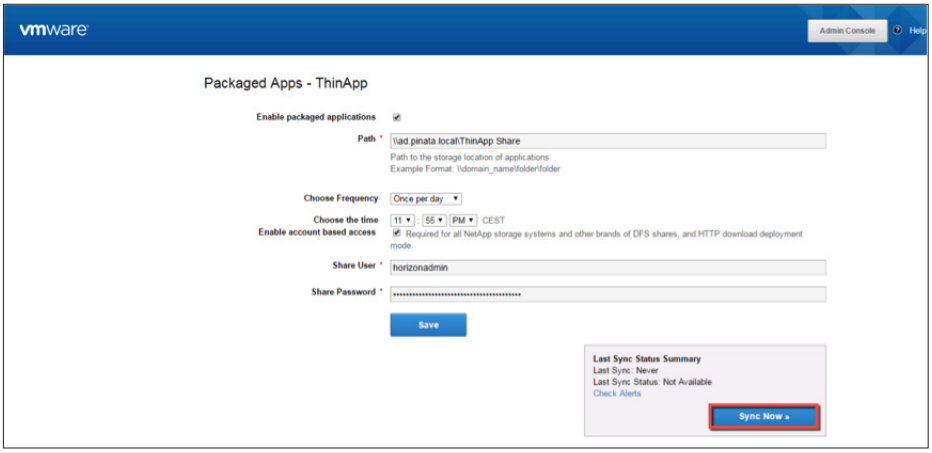

FIGURE 50: Sync the ThinApp Repository to VMware Identity Manager

8. Verify that the synchronization is successful and that you can see the ThinApp packages available to VMware Identity Manager. Click **OK**.

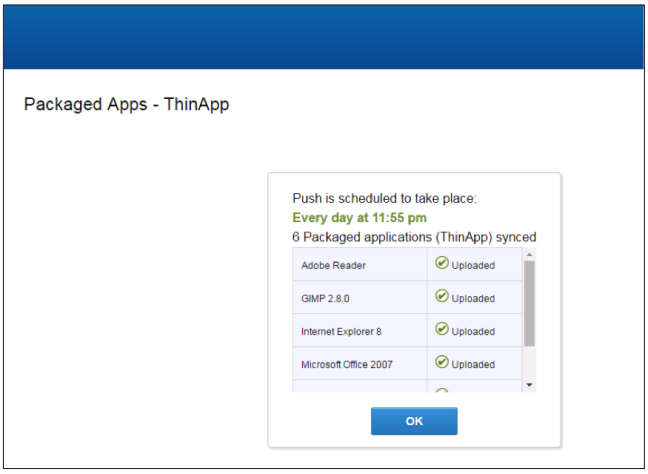

FIGURE 51: Successful ThinApp Synchronization

Entitle ThinApp Packages to Users

When you have integrated your VMware Identity Manager deployment with ThinApp, you can entitle available ThinApp packages to users and groups. This exercise describes how to entitle ThinApp packages to users.

1. Log in to the VMware Identity Manager administration console.
2. Click the **Catalog** tab.
3. (Optional) Click **Any Application Type** > **ThinApp Packages**. This filters the application list to show ThinApp packages only.

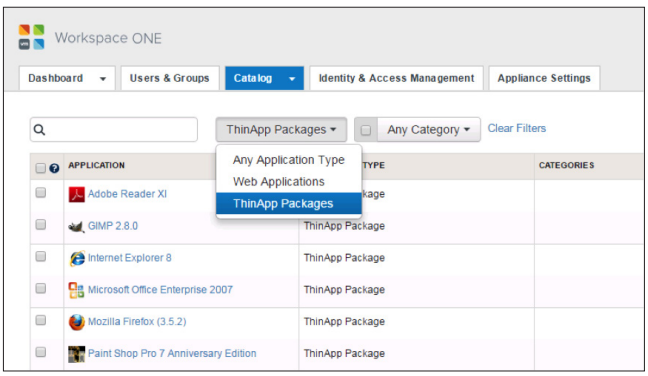

FIGURE 52: Filtering Application List to Show ThinApp Packages

4. Click an application to entitle a user. This example uses Adobe Reader.
5. Click **Add user entitlement**.

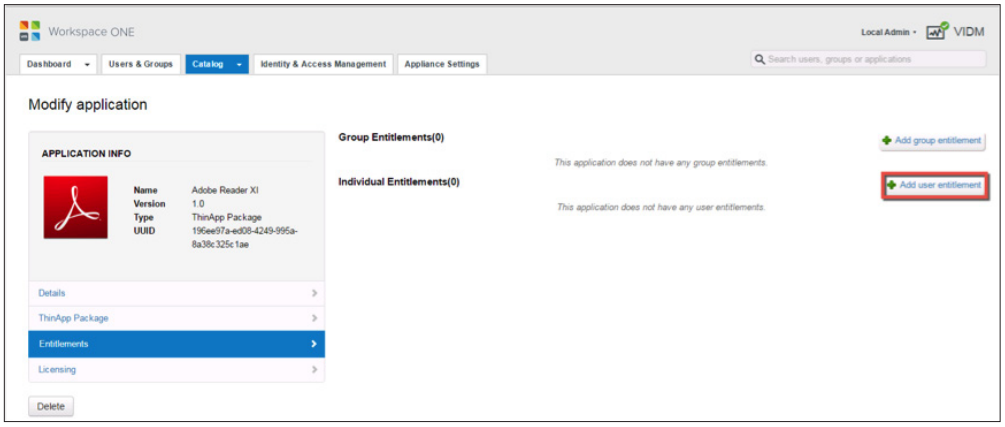

FIGURE 53: Adding User Entitlement

6. To add a user, enter the username. The user catalog is searched as soon as you start typing.
7. Select the entitlement method and click **Save**. VMware Identity Manager supports two methods.
   - Automatic entitlement gives the user immediate access to the application on the VMware Identity Manager page.
   - User-Activated entitlement lets the user add the application using the self-service application catalog.

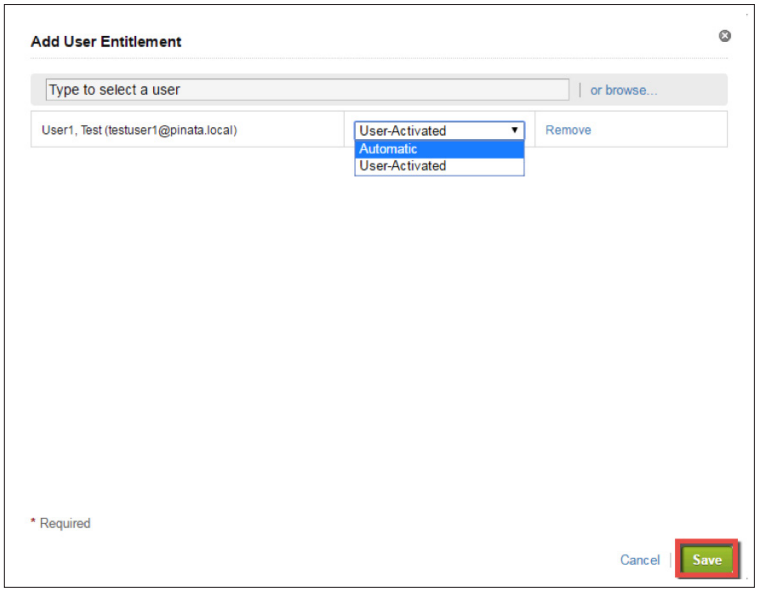

FIGURE 54: Selecting Entitlement Method

8. Click **Done** to activate the entitlement.

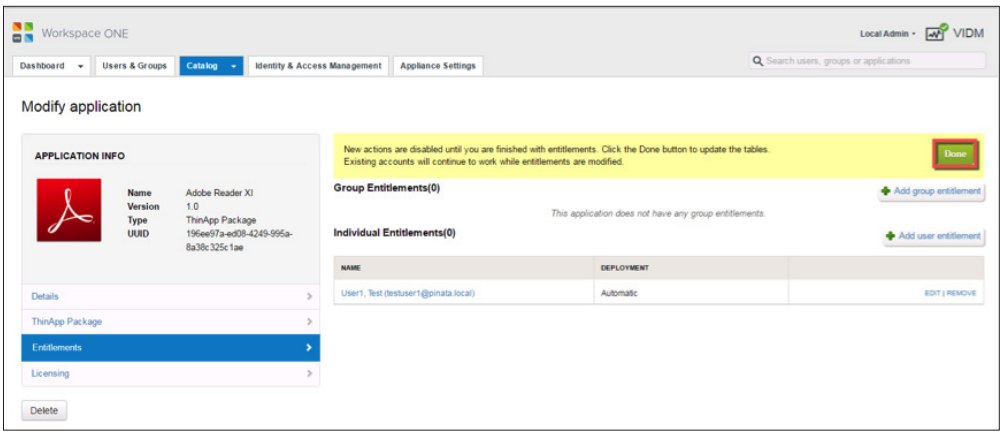

FIGURE 55: Activating New Entitlements

For more information, see *Entitle Users and Groups to ThinApp Packages* in the Setting Up Resources in the VMware Identity Manager guide.

Install the VMware Identity Manager Desktop Application on Client PCs

For ThinApp entitlement and deployment to work, each client must have the VMware Identity Manager Desktop application installed. VMware Identity Manager Desktop synchronizes the ThinApp packages to your Windows system, after verifying your entitlements.

If users attempt to start a VMware Identity Manager-enabled application without VMware Identity Manager Desktop installed, they receive an error message. The error message varies depending on from where the user tries to start the application.

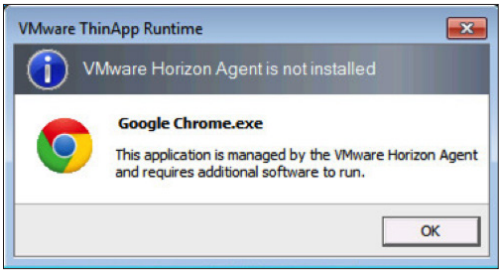

FIGURE 56: Error Message When User Starts an Application from a Desktop Shortcut

When VMware Identity Manager Desktop has been successfully installed, you can see its icon in the task bar.

FIGURE 57: VMware Identity Manager Desktop Icon

For more information, see *Installing Identity Manager Desktop* in Using VMware Identity Manager Desktop.

Start ThinApp Packages from VMware Identity Manager

When ThinApp and VMware Identity Manager are integrated and users have been entitled to ThinApp packages, the user can start these ThinApp packages from VMware Identity Manager.

1. Log in to the VMware Identity Manager user portal, start a Web browser, and go to `https://<IdentityManagerHostname>`. You will see the available applications. Applications that are user-activated can be added from the Catalog tab.

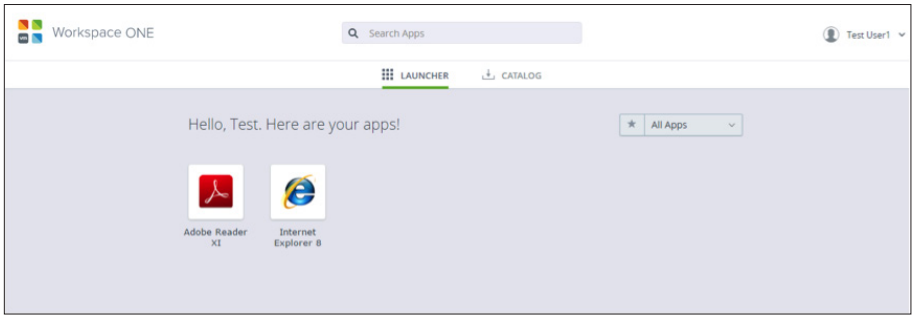

FIGURE 58: Available Applications

2. Click **Catalog** to see other applications that are available to this user.
3. To add another application, click **Add**.

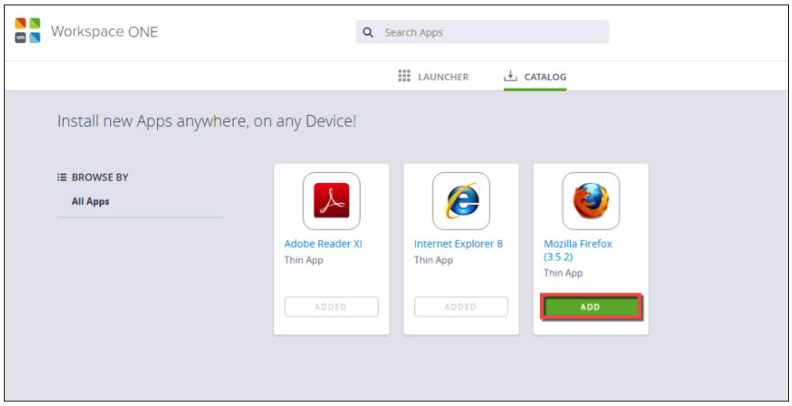

FIGURE 59: Adding Mozilla Firefox to Catalog

4. Click **Launcher** to see your updated application list.

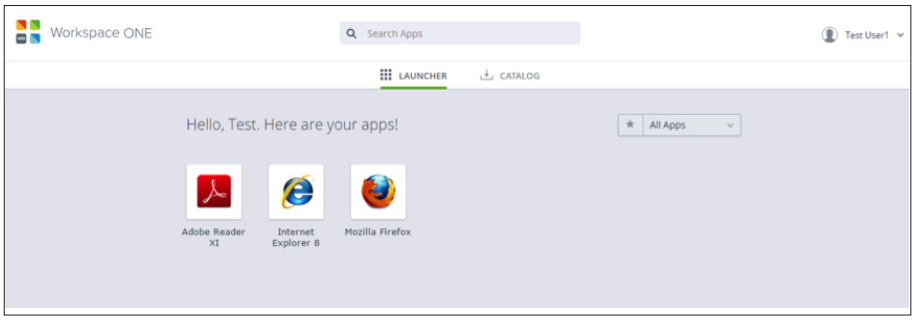

FIGURE 60: Updated Application List

5. Click an application to start it from VMware Identity Manager.

In this exercise, you configured a ThinApp repository for VMware Identity Manager, entitled ThinApp packages to a user, and started these entitled ThinApp packages from VMware Identity Manager.

Use ThinApp Packages with Mirage

Mirage is a layered-image management solution that separates a desktop, laptop, or virtual endpoint into logical divisions called *layers*. ThinApp can take advantage of the Mirage layering technology because packages are self-contained. Administrators can combine Mirage and ThinApp to deliver ThinApp packages to end users in one or more layers. Existing ThinApp packages can be used with Mirage, and you can install native applications on the same layer as the ThinApp packages, allowing support for most applications on a Windows desktop.

By combining these products, you also have the combined benefits of traditional local and remote ThinApp deployments. This provides greater manageability without sacrificing performance to a file share. More advantages are discussed in Mirage and ThinApp.

ThinApp packages installed to a Mirage layer must be MSI-based. Create a Mirage app layer on a provisioning machine, and install the ThinApp package at the prompt. Start your application to ensure that it is working as expected, and complete the app layer creation process. Assign the layer to specific machines in your environment.

For more information, see the VMware blog post Using ThinApp Virtual Applications in VMware Mirage App Layers.

Update ThinApp Packages

Application updates are necessary to provide end users new application features or to comply with administratively prescribed updates to software. You must decide whether the user or the administrator is responsible for updating an application. Users who self-update virtualized applications incorporate the changes directly into the application's sandbox, which can significantly increase its size. These sandbox settings can also interfere with future updates provided by the administrator.

There are six methods for an administrator to update a package. If this approach is taken, select the method most appropriate for the update you want to deploy.

- Recapture
- Sandbox Merge
- Post-Capture
- Project to Physical (P2P)
- AppLink
- Snapshots

Recapture

The recapture method goes through the complete Setup Capture again to incorporate the software and its updates between the Setup Capture prescan and postscan snapshots. The result of this process is a new package with the changes embedded. For example, if your original package was Microsoft OneNote, to create the updated package, install OneNote, apply the most recent Service Pack, and build a new package.

Sandbox Merge

A sandbox merge consolidates updates from a sandbox into an existing project directory. To use this method, start the virtualized application on a clean workstation and run the update, which places the new files, registry, and configuration changes into the sandbox of that computer. After the application is updated, use the ThinApp **`sbmerge`** utility to merge the changes from the sandbox into the existing project directory. Rebuild the package to incorporate the changes.

For more information, see *sbmerge.exe Commands* in the ThinApp User's Guide.

Post-Capture

The post-capture method involves manually placing folders in the appropriate directories of the capture, manually editing the registry files to include the changes, and editing the **`Package.ini`** file to change configuration settings. Use this method when you definitively know which files or registry changes you want to make. This method does not require using Setup Capture, but you must rebuild the package with the **`build.bat`** file to incorporate the changes.

Project to Physical (P2P)

For a large project, such as Microsoft Office, it can be difficult to keep track of all the modifications made. If you recapture Microsoft Office, you need to reinstall the existing components and reapply the existing settings, along with the updates. A benefit of P2P is that, after you have taken a prescan and extracted a project, all you need to do is apply the updates and take a postscan.

For more information, see *Use snapshot.exe Utility to Extract a ThinApp Project* in the ThinApp User's Guide.

AppLink

AppLink was originally designed to handle dependencies, however, you can also use it to update a ThinApp package. For example, you can update a configuration file or add plug-ins or components to an existing package. The advantage of AppLink is its ability to handle conflicts. For example, if the same file exists in two packages, the last loaded package takes precedence. The initial or *parent* package is always loaded first, followed by the AppLink or *child* package. In the case of a conflict, the file in the child package is used.

For example, the parent package has the following configuration file:

`C:\Program Files\ApplicationName\MyConfiguration.ini.`

To update the configuration file, create an empty project with the updated file in the same location (`C:\Program Files\ApplicationName\MyConfiguration.ini`) and AppLink it to the parent package.

Note: If the application has already been deployed to the user and it originally did not have AppLink enabled, then you must redeploy the entire package. This would negate the need for AppLink as the changes could be incorporated into the new version. Alternatively, you could apply the AppLink update with GPOs.

You can deploy more complex updates with AppLink. However, it is not suitable for security and service pack updates. AppLink works, but if you use `OptionalAppLinks` and the child package is not present, your environment might be unpatched.

Snapshots

You can use snapshots to update your ThinApp packages. It is best practice to take a snapshot of your virtual machine prior to a postscan. At this point, you have run prescan and installed the native application. When an update is needed, you can revert to this snapshot, update the application, and take another postscan. This eliminates the need to run through the entire Setup Capture process again.

Update ThinApp Packages in View

When considering update methods for ThinApp packages on VDI desktops, it is important to remember that a virtual desktop used with ThinApp is a Windows machine. For this reason, you can update ThinApp packages on a VDI desktop in the same manner as you would on any Windows machine.

If ThinApp packages are part of a linked-clone master image, you can update the packages on the master image and then recompose the linked-clone pool.

Update ThinApp Packages in VMware Identity Manager

Each build of a ThinApp package enabled for VMware Identity Manager generates a unique GUID for the package through the `AppID Package.ini` parameter setting. ThinApp 4.7.2 and later provides a new update mechanism for ThinApp packages in VMware Identity Manager.

VMware Identity Manager uses the new `Package.ini` parameter `VersionID`, in conjunction with the package identifier `AppID`, to determine if a ThinApp package is an updated package. `VersionID` specifies the version of the ThinApp application. The default value is `1`. For an updated package to be recognized by VMware Identity Manager, it must have the same `AppID` as the previous version, and the `VersionID` must be incremented. This method cannot be used for virtual applications outside of VMware Identity Manager.

When VMware Identity Manager scans the ThinApp repository and finds an application with the same `AppID` as another application, it compares `VersionID` values. The package with the highest `VersionID` is used. If an updated package is detected, VMware Identity Manager applies the previous user entitlements to the package, so you do not have to re-entitle users. Shortcuts on the endpoints are changed to point to the new package.

Entitlements pertain to a specific application ID. Therefore, neither AppSync nor side-by-side updating of ThinApp packages is supported in VMware Identity Manager.

For more information, see the following resources.

- *Updating Managed ThinApp Packages After Deployment in VMware Identity Manager* in Setting Up Resources in VMware Identity Manager
- VMware blog post Using Horizon Workspace/Application Manager to update ThinApp packages

Deploy Updates

After application changes have been incorporated into an updated package, you can select from five methods to deploy it.

- Package Replacement
- Side-by-Side Update
- AppSync
- MSI
- Package Management Using Group Policy

Package Replacement

You can use the package replacement update method for either a remote or local deployment. If you have created an updated package and there is a time when the application is not in use, you can replace the original Primary Data Container (EXE or DAT) with the updated PDC. Make sure that the filename is exactly the same, because users may depend on previously created shortcuts to start applications.

Exercise 9: Side-by-Side Update

The side-by-side method is most effective for ThinApp packages on file shares, because you only need to update one instance of the package for all users. Users experience no downtime, and administrators do not have to wait for the application to no longer be in use. The updated package runs when the user restarts the application. An additional benefit is that, if necessary, you can roll back to a prior version of the application by giving the version you want a higher integer suffix in its filename extension.

You can use the side-by-side method for updating ThinApp application packages for either a remote or local deployment. It does not require user or application downtime. This method places the updated application package in the same directory as the original application package and gives the updated PDC an integer filename extension, such as `.1`. Subsequent updates can be placed in the same directory with incremented filename extensions, such as `.2` or `.3`.

The following steps show how to use the side-by-side method for packages on file shares. You can use the same method for local packages. These steps are also demonstrated by this silent video: ThinApp SxS Updating.mp4.

1. Create a package of an application to update in this exercise. For example, Mozilla Firefox 30.0 built as a ThinApp package named **`Mozilla Firefox.exe`**.
2. Create a second package with the updated application version that you want to deploy. For example, Mozilla Firefox 35.0.1, built as a ThinApp package named **`Mozilla Firefox Update.exe`**.
3. Copy the two packages into the same directory on a file share.
4. As a user, start the initial package, **`Mozilla Firefox.exe`** and note the version number. It is not necessary to close the application.

FIGURE 61: Using the Side-by-Side Update Method

5. Rename the updated application package from **`Mozilla Firefox Update.exe`** to **`Mozilla Firefox.exe.1`**. You do not need to rename any entry points (in this example, there are none)—rename only the PDC. Make sure that the filename extensions are visible in Windows.

   Note: If an ALT file is present, it must also be renamed, and it follows a different naming convention. For example, **`Mozilla Firefox.alt`** is renamed to **`Mozilla Firefox.1.alt`**.

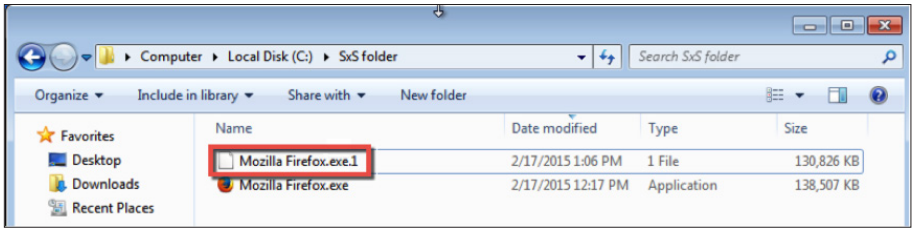

FIGURE 62: Confirm Filename Extension in Windows

6. Restart the application. Check that the updated version of the application opened.

FIGURE 63: Check That New Application Starts

Notes

- If you have been double-clicking **`Mozilla Firefox.exe`** to start Firefox, continue to do so. ThinApp itself will detect the existence of the **`Mozilla Firefox.exe.1`** file and automatically use it instead. Trying to start the updated Firefox by double-clicking **`Mozilla Firefox.exe.1`** will not be successful.
- When you know that nobody is using an application that you have updated in this fashion, you can delete the original file, and rename the updated file to remove the integer extension from its filename.

In this exercise, you updated Firefox by giving the updated package an integer filename extension, put it in the same directory as the original package, and confirmed that when you restarted Firefox the updated version was run.

Exercise 10: AppSync

AppSync provides an easy-to-administer method of updating packages inside or outside the corporate network. AppSync initiates the pull of any available differential update packages from a central Web server or UNC location at regular intervals. You can configure these settings in the **`Package.ini`** file.

AppSync offers a hybrid approach to deploying applications. With AppSync, a *master image* of the package is stored on a network share. After initial application installation, you can manage and update this image centrally. Each time you update the application, it is copied to every endpoint. Benefits of remote and local deployment are achieved, simultaneously. Each endpoint starts the application locally, resulting in faster performance. And just like remote deployment, AppSync, run on each endpoint, pulls its updates from a single network location. But unlike normal remote deployment, if there is no network connection end users can still use the application, even if it is not the most recent version.

For a demonstration of the AppSync feature, refer to the ThinApp AppSync with Multiple Entry Points video.

Note: You can reuse the packages created in the side-by-side example and skip to Step 3.

1. Create a package of an application to update in this exercise. For example, Mozilla Firefox 30.0 built as a ThinApp package named **`Mozilla Firefox.exe`**.
2. Create a second package with the updated application version that you want to deploy. For example, Mozilla Firefox 35.0.1, built as a ThinApp package named **`Mozilla Firefox Update.exe`**.
3. In the first package directory, edit the **`Package.ini`** file to have AppSync point to a URL or UNC location for the update server. Some examples:
   - `AppSyncURL=https://<site.com>/<path>/<primary_data_container_filename>`
   - `AppSyncURL=file://<server>/<share>/<path>/<primary_data_container_filename>`
4. In the **`Package.ini`** file of both packages, set the PDC to the same name. The PDC includes a *ReadOnlyData* line. The PDC name is in square brackets at the beginning of the entry. For AppSync to work, the PDC names in the original and updated packages must be identical.

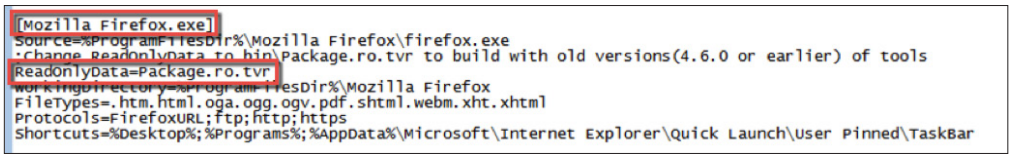

FIGURE 64: PDC Entry in the Package.ini File

5. If you change the PDC name in **`Package.ini`** for either or both packages, change the Shortcut parameter value for all entries in the **`Package.ini`** file to the new PDC name.
6. Rebuild both the old and updated packages to incorporate the changed **`Package.ini`** settings.
7. Deploy the initial package to a desktop using a local deployment method.
8. Place the updated package in the URL or UNC location specified in the **`AppSyncURL`** parameter.
9. Start the application. By default, the update is downloaded and applied on the next start of the application.
10. Close the application and restart it to confirm that the update was successful.

In this exercise, you updated a package using the AppSync feature. You confirmed that the original version of the application was initially started, and the updated version started when the application was restarted.

AppSync is typically used in a local network environment as data is transferred over HTTP/HTTPS.

For more information, see *Configuring Application Updates with the Application Sync Utility* in the ThinApp Package.ini Parameters Reference Guide.

MSI

You can use MSI files to deploy both new and updated ThinApp packages. If you are already using MSI files to deploy new applications to users, you are probably using some form of Electronic Software Distribution (ESD). You can use existing processes to deliver updated MSI-based ThinApp packages to end users.

To update a package, the following `Package.ini` parameters are required.

- **MSIProductVersion** – Specifies the product version number used by the MSI database for version control. This value does not relate to the application version or the ThinApp version. ThinApp assigns a default version of 1.0. If you create an updated package and change this parameter to 2.0, ThinApp uninstalls the 1.0 version and installs the 2.0 version of the package.
- **MSIUpgradeCode** – Specifies the code used by the MSI database that facilitates updates. When two packages, such as version 1.0 and version 2.0, have the same upgrade code, the MSI installer detects this link, uninstalls the earlier package, and installs the updated package. The capture process generates a random upgrade code based on the inventory name. To ensure that the MSI database versions have the same upgrade code, keep the same inventory name across different versions of the MSI-based project.

For more information, see the VMware blog post Upgrade a deployed ThinApp package with the help of MSI.

Package Management Using Group Policies

ThinApp 5.1 introduces a package management feature that allows administrators to dynamically reconfigure the attributes of the following aspects of a ThinApp package at runtime.

- AppLink
- AppSync
- Entry-point shortcuts
- ThinDirect

To manage a ThinApp package, an associated group policy must be defined using the inventory name of that package. You can also define group policies for a specific set of client machines using group policy attributes.

ThinApp 5.1 provides template files that you can use to enable package management in your environment. If a package is managed by a group policy, ThinApp gives precedence to the group policy over the `Package.ini` settings. For example, if you have specified AppLink settings in `Package.ini` and you also have a group policy defined for AppLink with different settings, the settings defined in the group policy are applied to the ThinApp package at runtime.

If you are using GPOs in your environment, you can enable AppSync or AppLink in existing ThinApp packages, even if they were built without these features enabled. You do not have to edit `Package.ini` and rebuild; You can enable the features using only GPOs. This feature is available only with ThinApp 5.1 or later.

For more information, see Package Management.

Exercise 11: Use AppLink to Combine ThinApp Packages

AppLink connects dependent application packages at runtime, allowing you to create relationships between packages without using Setup Capture to build the needed components into a single package. In addition, component packages are often more efficiently built, deployed, and updated separately.

You can specify a required or optional AppLink in `Package.ini`. If you specify a required AppLink, the parent package does not start if it cannot connect to the child package. If you specify an optional AppLink, the parent package starts even if it cannot connect to the child package.

Both required and optional AppLinks can use nested links between multiple packages.

If you specify an optional AppLink, you can create a link to a directory using the * wildcard to establish links to all packages in that directory. If you specify a required AppLink, you must explicitly name each package to link.

Here are some scenarios for creating links between packages using AppLink.

- Link runtime components, such as .NET, JRE, or ODBC drivers, with dependent applications. For example, you can link .NET to an application even if the local machine for the application has a different version of .NET.
- Package and deploy application-specific components and plug-ins separately from the base application. For example, you might separate Adobe Flash Player or Adobe Reader from a base Firefox application and link the components.

For more information, see *Application Link Updates* in the ThinApp User's Guide.

Link Required Packages

The following exercise guides you through setting the `RequiredAppLinks` parameter and attempting to start a package both with and without a required child package.

1. Create an application package. For example, build a ThinApp package with Firefox and named `Mozilla Firefox.exe`.
2. Create a second package with the component that you want to link. For example, build a ThinApp package with Flash Player and named `AdobeFlashPlayer.dat`.
3. In the Firefox package, open the `Package.ini` file and set the `RequiredAppLinks` parameter in the `[Build Options]` section to the following:

```
;OptionalAppLinks=plugins\*.exe
RequiredAppLinks=C:\AppLinks\AdobeFlashPlayer.dat
```

FIGURE 65: RequiredAppLinks Parameter in Package.ini

Note: Whether you select `OptionalAppLinks` or `RequiredAppLinks`, it is the PDC that is specified. The PDC can be either an EXE or DAT file.

In this example, the AppLink package is stored locally on the C: drive. You can specify a file share in the format: `\\servername\sharename\AdobeFlashPlayer.dat`

4. Place the Mozilla Firefox package in a local directory. You can also place both packages on a file share and change the `RequiredAppLinks` parameter accordingly.
5. Start Firefox and navigate to https://www.adobe.com/software/flash/about to verify Flash function.

Now, attempt to start Firefox when the `AdobeFlashPlayer.dat` file is missing from the specified directory.

6. Remove `AdobeFlashPayer.dat` from `C:\AppLinks`.
7. Start Firefox and note the error.

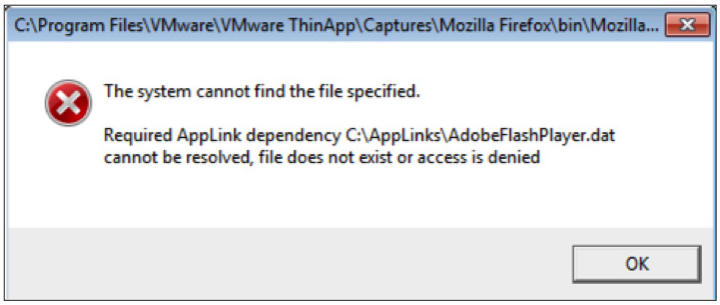

FIGURE 66: Error When Required AppLink Is Unavailable

Link Optional Packages

The following exercise guides you through setting the `OptionalAppLinks` parameter and attempting to start a package both with and without the optional child package.

Use the Firefox and Flash Player packages created in the previous exercise.

1. Place `AdobeFlashPlayer.dat` in `C:\AppLinks`.
2. In the Firefox package, open the `Package.ini` file and set the `OptionalAppLinks` parameter in the `[Build Options]` section to the following:

```
OptionalAppLinks=C:\AppLinks\AdobeFlashPlayer.dat
```

FIGURE 67: OptionalAppLinks Parameter in Package.ini

> **Note:** If you have completed the previous exercise, you will need to comment out the RequiredAppLinks parameter.

3. Place the Mozilla Firefox package in a local directory. You can also place both packages on a file share and change the `OptionalAppLinks` parameter accordingly.
4. Start Firefox and navigate to https://www.adobe.com/software/flash/about to verify Flash function.

Next, we will attempt to start Firefox when the `AdobeFlashPlayer.dat` file is missing from the specified directory.

5. Remove `AdobeFlashPayer.dat` from `C:\AppLinks`.
6. Start Firefox. The application starts even though the child package is not available. (However, if you navigate to the Adobe site, you will notice that Flash is not available.)

For more information, see Appendix C: Optional Versus Required AppLinks.

Exercise 12: Configure AppLink Using Group Policy Objects

In previous versions of ThinApp, it was possible to control ThinDirect using GPOs. With ThinApp 5.1, you can now use GPOs to also manage AppLink, AppSync, and entry point shortcuts for deployed ThinApp packages.

This section demonstrates dynamic management of AppLink using GPOs. It includes the following exercises.

- Customize the Template Files for Firefox
- Place the Administrative Template Files on the Domain Controller
- Create and Configure a GPO for Firefox
- Link the GPO to a Domain
- Update Client Machines

The following exercises must be completed in the order presented.

For these exercises, you need

- Access to a domain controller
- A Firefox ThinApp package
- An Adobe Flash Player ThinApp package

The steps in these exercises pertain to Firefox, but they can be applied to any application you want to control using GPOs. You can verify steps needed in the **README.TXT** file located in the Policy folder in the ThinApp installation directory.

Customize the Template Files for Firefox

ThinApp 5.1 includes group policy Administrative Template (ADMX and ADML) files that you can customize for any ThinApp application.

Before you begin, confirm that AppLink has not yet been configured for the Firefox or Flash Player ThinApp packages.

1. On a clean machine, start virtual Firefox and confirm that Flash Player is not available by going to http://www.adobe.com/software/flash/about. Note the error message: "A plugin is needed to display this content."

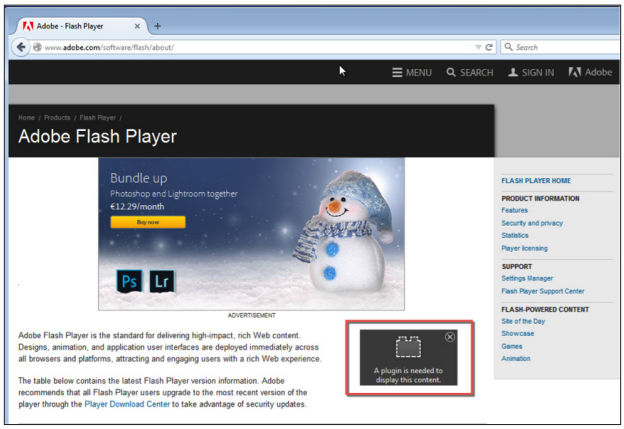

FIGURE 68: Firefox Without Flash Plug-In

The next step is to customize the existing administrative template files for the Firefox application.

2. Go to the ThinApp directory and browse to the `Policy` folder, which contains the administrative templates used to manage AppLink with GPOs.

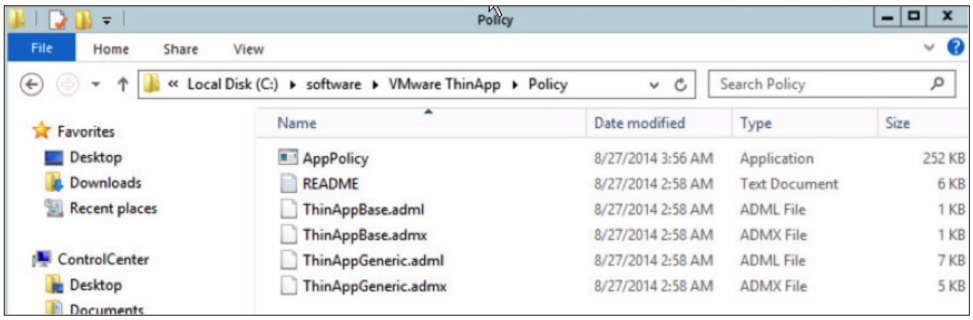

FIGURE 69: Administrative Templates in the Policy Folder

3. Customize the template files for Firefox.
   a. Note the inventory name of the ThinApp application. In this example, the inventory name is `Mozilla Firefox`.

      If you do not know the inventory name, you can query it by running the command

      `AppPolicy.exe /s <path to the package PDC file>`

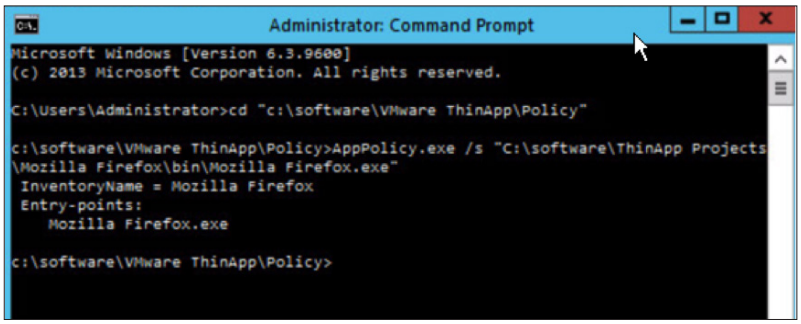

FIGURE 70: Running AppPolicy.exe to Get the Correct Inventory Name

b. The generic template files (`ThinAppGeneric.admx/adml`) contain settings for a generic application. To customize these files for Firefox, use its inventory name and run the command

   `AppPolicy.exe /c "Mozilla Firefox"`

   `AppPolicy.exe` creates a new set of application-specific files named `ThinApp_MozillaFirefox.admx` and `ThinApp_MozillaFirefox.adml`.

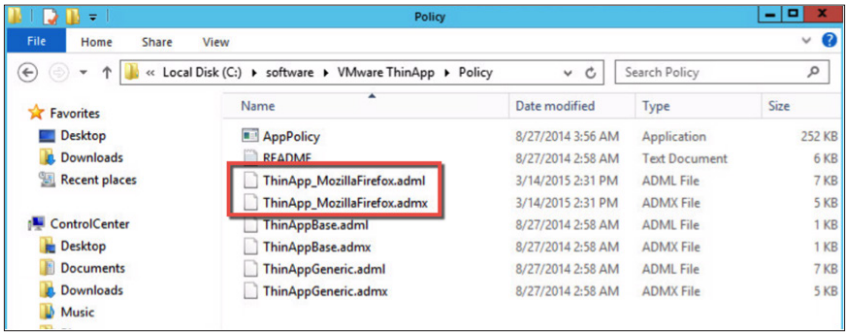

FIGURE 71: ADMX and ADML Files for Mozilla Firefox

Note: All policy settings in the application-specific template files target an Active Directory *Machine* by default rather than an Active Directory *User*. These settings are applied as computer-level settings and appear under **Computer Configuration** > **Policies** > **Administrative Templates** in the Group Policy Management editor. To target the `User`, open the ADMX file and replace all occurrences of the line `class="Machine"` with `class="User"`.

Place the Administrative Template Files on the Domain Controller

When the generic template files have been customized for the ThinApp application, the next step is to place the customized template files, along with the base template files, on the domain controller.

1. Move the application-specific template files to the `PolicyDefinitions` folder. You need to perform these steps for each ThinApp package that you want to control using GPOs.
   a. Move `ThinApp_MozillaFirefox.admx` to `%WINDIR%\PolicyDefinitions`.
   b. Move `ThinApp_MozillaFirefox.adml` to `%WINDIR%\PolicyDefinitions\en-US`.

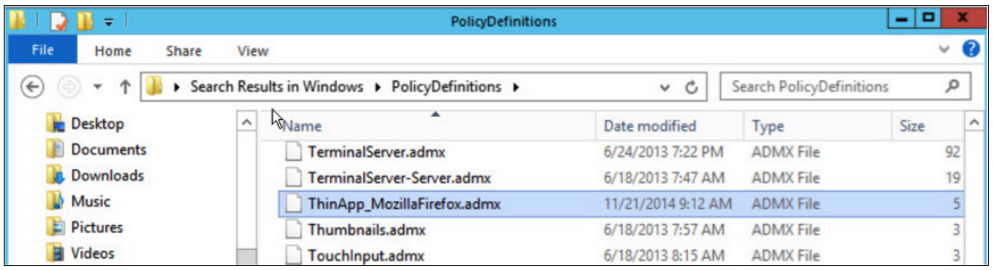

FIGURE 72: Mozilla Firefox ADMX File in PolicyDefinitions Folder

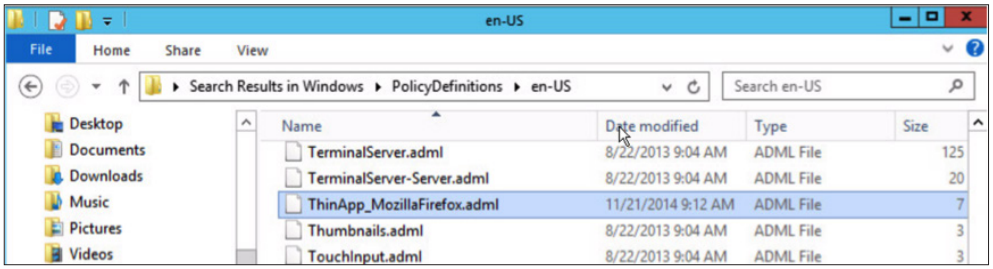

FIGURE 73: Mozilla Firefox ADML File in en-US Folder

2. Copy the base template files to the domain controller's `PolicyDefinitions` folder. Go to the `VMware ThinApp` directory and browse to the folder. You need to copy these files only once, even if you have more than one ThinApp application to control using GPOs.
   a. Copy `ThinAppBase.admx` to `%WINDIR%\PolicyDefinitions`.
   b. Copy `ThinAppBase.adml` to `%WINDIR%\PolicyDefinitions\en-US`.

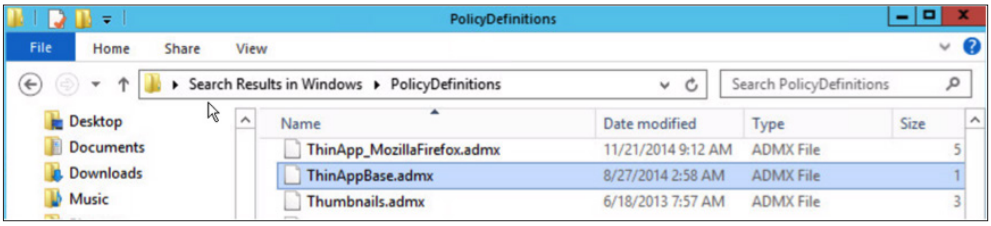

FIGURE 74: ThinAppBase ADMX File in PolicyDefinitions Folder

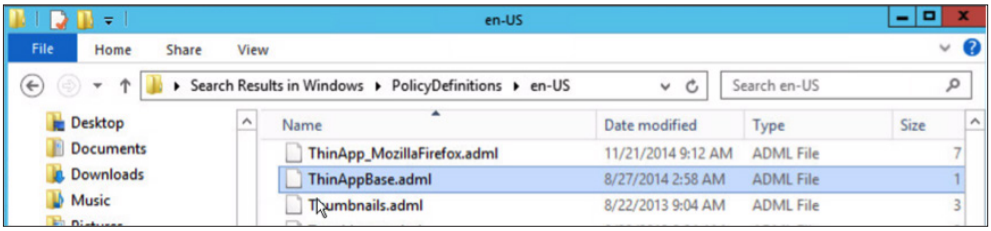

FIGURE 75: ThinAppBase ADML File in en-US Folder

Create and Configure a GPO for Firefox

After you have copied the base template files and the application-specific template files to the domain controller, you can create a GPO for your ThinApp application and configure the settings for this GPO.

1. From a command line, run `gpmc.msc` to open the Group Policy Management Console (GPMC).
   a. Expand **Group Policy Management** > **Forest** > **Domains**.
   b. Under your domain, right-click **Group Policy Objects**, and select **New**.

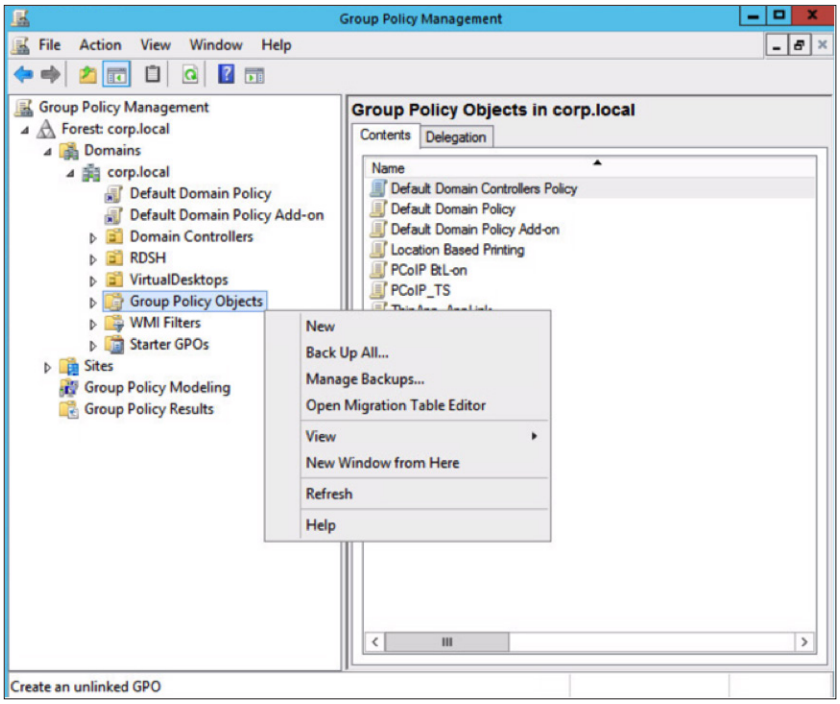

FIGURE 76: Creating a New Group Policy Object

c. Enter a name for this GPO object, for example, `GPO-ThinApp-MozillaFirefox`.

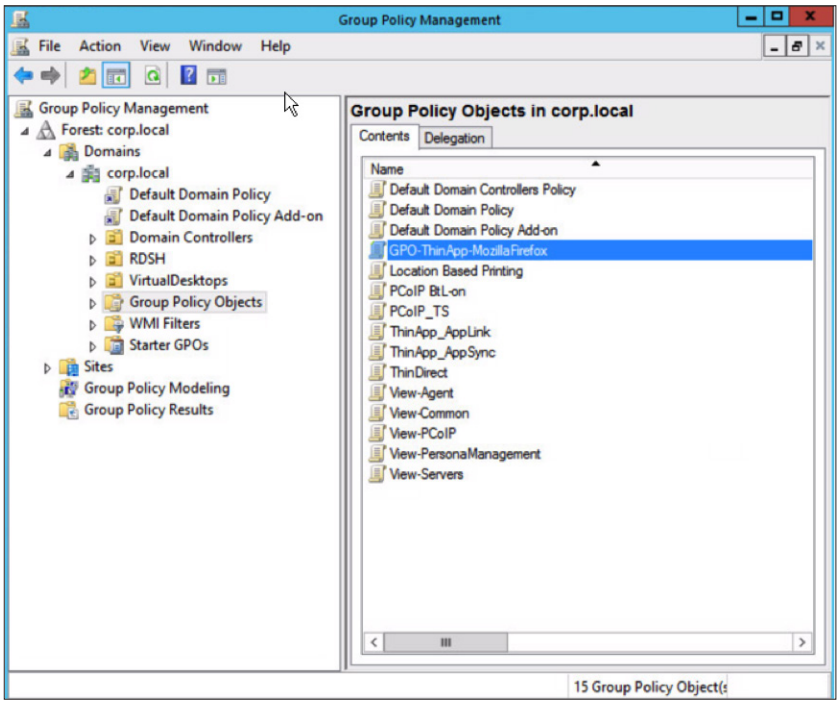

FIGURE 77: Newly Created GPO: GPO-ThinApp-MozillaFirefox

2. Configure the new GPO.
   a. Right-click **GPO-ThinApp-MozillaFirefox** and select **Edit**. The Group Policy Management Editor opens.

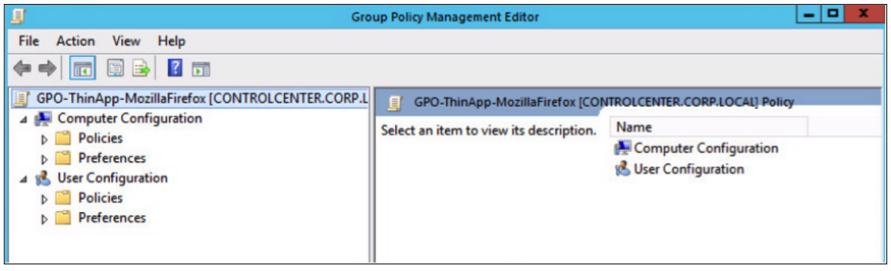

FIGURE 78: Group Policy Management Editor for GPO-ThinApp-MozillaFirefox

   b. Because this GPO targets machines, expand **Computer Configuration**. (If your GPO targets users, expand **User Configuration**. If your GPO targets both machines and users, expand both **Computer Configuration** and **User Configuration**.)

c. Expand **Policies** > **Administrative Templates** > **VMware ThinApp Management**.

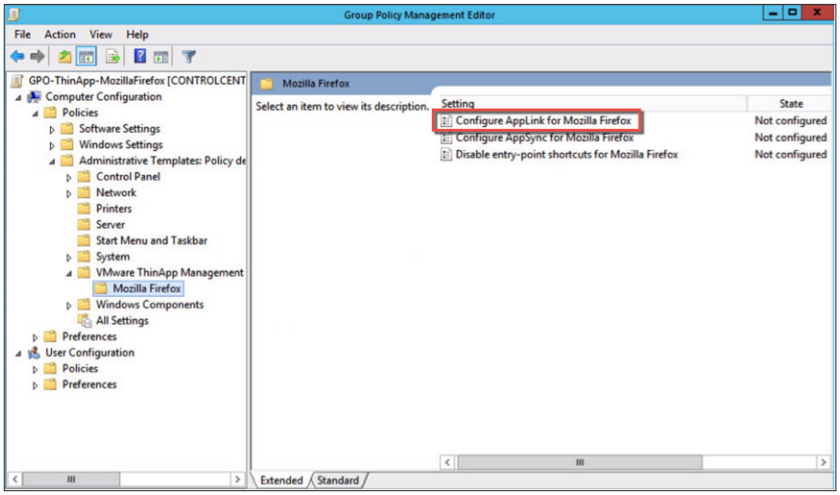

FIGURE 79: Configure the GPO in the Group Policy Management Editor

d. Click **Mozilla Firefox** to display the AppLink policy settings.

e. Select **Enabled** to enable this GPO. AppLink is performed by the ThinApp runtime even if no AppLink settings are defined in `Package.ini`.

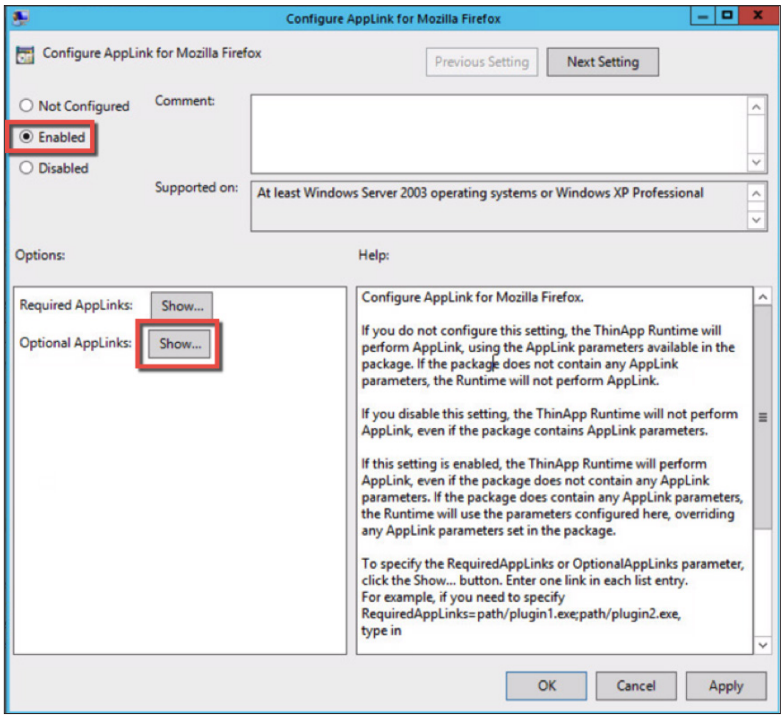

FIGURE 80: Enable AppLink in Configure AppLink for Mozilla Firefox

f. Next to Optional AppLinks, click **Show**. Specify the full path to the Adobe Flash Player ThinApp package. This path is fully customizable, and you can also specify multiple locations. Paths relative to the configured application are allowed. In this example, the `Adobe.exe` file is in a subfolder of the Mozilla Firefox directory called `plugins`.

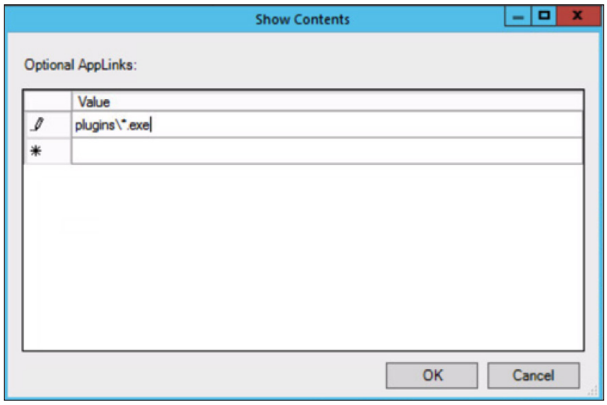

FIGURE 81: Specify Path to Adobe Plug-In

g. Due to a limitation in the Group Policy Editor, you must enter a value for both Required AppLinks and Optional AppLinks. If you try to save your settings without adding entries to each list, you are prompted for a value.

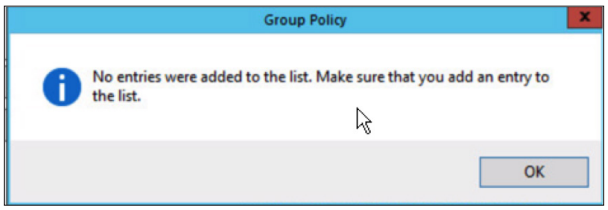

FIGURE 82: Error When Attempting to Save GPO Without List Entries

If an entry has no required values, click **Show** next to it (see Figure 80) and enter a space. Read the **Help** box for more information.

h. Click **Apply** and **OK** to complete the AppLink configuration.

Link the GPO to a Domain

When the Mozilla Firefox GPO has been created and configured, the next step is to link this GPO to the domain. It is also possible to link the GPO to a site or an organizational unit (OU).

1. In the Group Policy Management console, right-click your domain under Domains and select **Link an Existing GPO**.

   The Select GPO dialog box opens.

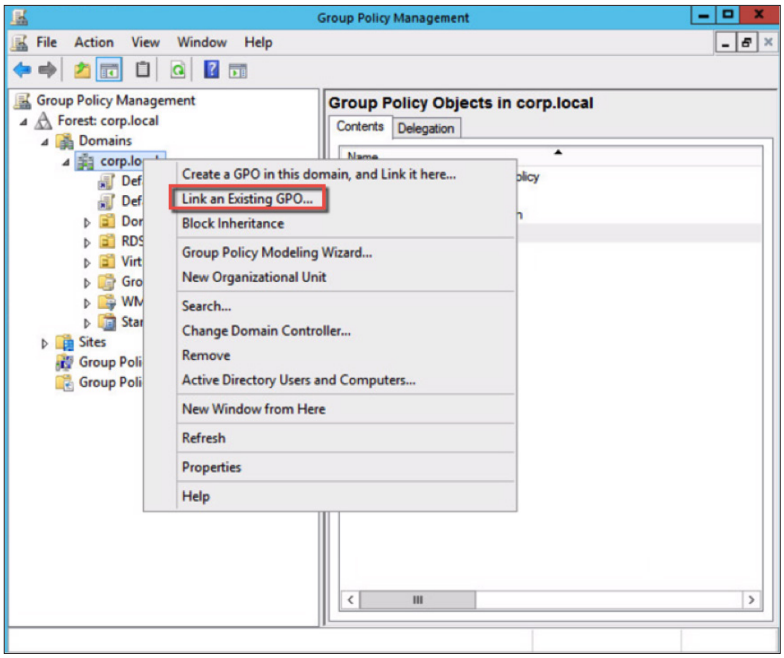

FIGURE 83: Link an Existing GPO in Group Policy Management

2. Under Group Policy objects, select **GPO-ThinApp-MozillaFirefox** and click **OK**.

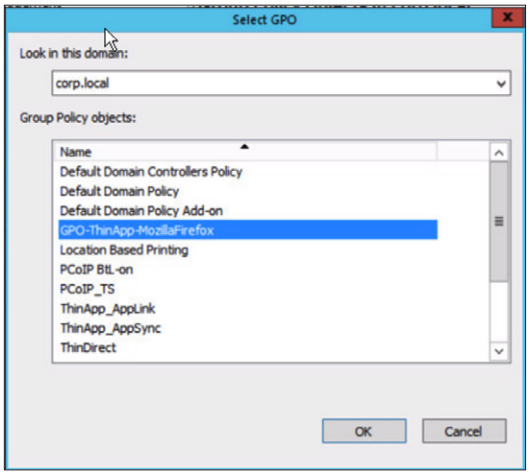

FIGURE 84: Select the GPO

Update Client Machines

The final step is to update the policy on the client machine and verify that Adobe Flash content can be viewed on Firefox.

1. On the client machine, run `gpupdate /force` from a command line to update the policy. You can also reboot the machine to update the policy.

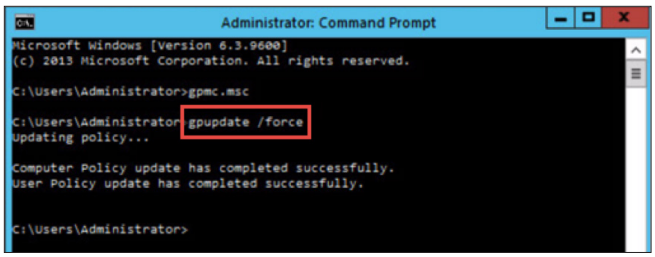

FIGURE 85: Updating the Group Policy on the Client Machine

2. Verify that Firefox has been AppLinked to Flash Player. Start Firefox and navigate to http://www.adobe.com/software/flash/about. If AppLink is successful, you will see video content playing.

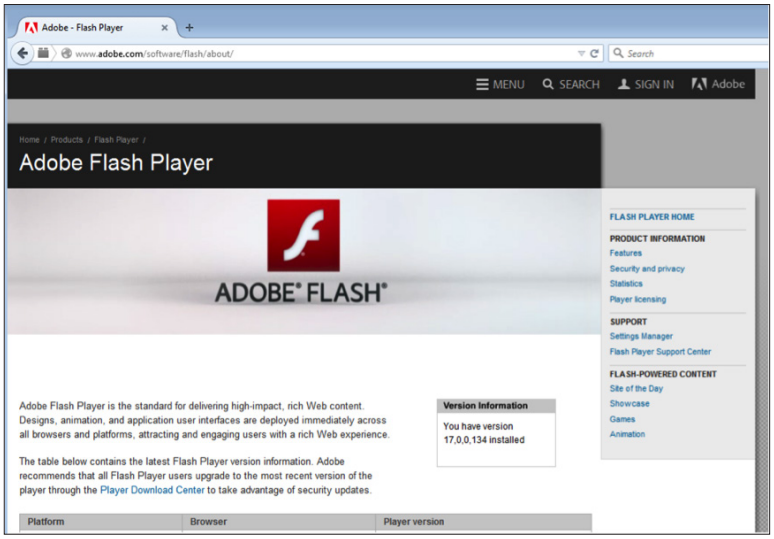

FIGURE 86: Firefox Linked with Flash Player

In this exercise, you have customized template files for Firefox and created and configured an AppLink GPO to link Firefox with Flash Player.

Summary

The topics covered in this guide included the ongoing key features and benefits of using ThinApp for application virtualization and isolation. The guide provided an overview of ThinApp functionality, architecture, and components. Exercises were provided to guide you through packaging, updating, and deploying applications.

Integration with external products including View, VMware Identity Manager, App Volumes, and Mirage were described and exercises were provided to help you explore and evaluate the benefits of ThinApp functionality.

Appendix A: Licensing

The ThinApp Suite contains the following components.

- VMware ThinApp packager and 50 client licenses
- VMware Workstation

Additional client licenses can be purchased in groups of 100.

The ThinApp Suite offers the following licensing options.

- Per endpoint
- Per concurrent user
- Per named user

What you need to get started:

- VMware ThinApp software and product key
- Clean installation of a Windows operating system on a dedicated physical or virtual capture machine
- Application installer file for the application you want to virtualize
- (Optional) View in VMware Horizon 7 environment

To obtain the evaluation software and product key for 50 clients, go to the VMware ThinApp Product Evaluation Center.

Your trial includes

- VMware ThinApp packager and 50 client licenses
- VMware Workstation

The trial licenses for ThinApp and Workstation are valid for 60 days.

For more information, see the ThinApp Community Web site. For HTML- and PDF-based product documentation, see the ThinApp documentation.

Appendix B: Deploying ThinApp Packages

Determining remote or local deployment depends on the delivery of the package. The same package can be stored remotely on a file share or locally on the end user's desktop with either mode. Each option provides different benefits and has different requirements.

You can also combine the two methods by deploying some applications remotely and others locally. Determine the appropriate deployment mode for each application.

In addition, AppSync offers a hybrid approach to deployment. It provides a one-to-many remote version of the application that can be managed and updated in a single location, and also results in a local copy on each endpoint. The following sections describe the two methods of providing virtualized applications to end users. For more information, see Updating ThinApp Packages.

Remote Deployment

Remote deployment allows the application to be centrally stored and accessed by multiple users. Remote deployment is a one-to-many model that provides centralized management and updating of the application package, providing it to multiple end users for execution through a Windows desktop shortcut.

Remote deployment is often the best option for environments that are centralized and where desktops are always online. In remote deployment, the application is started from a shortcut on the desktop or Start menu and run from a remote location.

Requirements

The user must always have access to the central network location where the ThinApp package resides.

Recommendations

The storage location that hosts the application must be highly available so that any brief, unscheduled downtime of either the host or storage device does not impact the environment. The use of SAN, DFS, and file-replication technologies is sufficient to make the file share highly available and fault tolerant.

Network bandwidth between the endpoint and the central network location must be robust. A virtualized application uses standard SMB protocol. The amount of network traffic varies based on the application and the functions used by the end user.

Benefits

Centralized management is the primary benefit of remote deployment. The one-to-many model of deploying an application to one location for many users is an efficient and effective model for application delivery. Providing access to the application merely involves placing a shortcut to the application on each desktop, which can be automated with the Thinreg utility in a Windows logon script.

The application package, which can be large in size, does not have to be delivered to the endpoint, so you do not have to transfer large files across a network or integrate with a deployment mechanism to distribute them. Additionally, there is no local disk footprint on the endpoint, because the application is streamed into memory.

For users who access applications from multiple endpoints, remote deployment provides a single point of administration and a consistent user experience across endpoints.

Drawbacks

Placing ThinApp packages on a file share can reduce performance. However, combining ThinApp with other products can mitigate this. See How ThinApp Complements Other VMware Products.

Local Deployment

A locally deployed application package is copied in its entirety to the end user's system. Users start the application from this local copy, which allows for offline application use.

Local deployment involves distributing the virtualized application package to the end user's virtual or physical desktop. The actual location of the package can be on the local file system or removable media. In this distributed model, each endpoint can run the application regardless of network connectivity. Endpoints that are occasionally or always offline require local deployment.

Requirements

Distributing the package to a local endpoint is required. A number of options exist: Active Directory–based publishing using group policies, third-party software deployment solutions, and custom scripted mechanisms. Users who are occasionally offline must have all applications and components deployed before working offline. Subsequent application deployment and updates are subject to network availability or other mechanisms, such as providing removable media with updates.

Recommendations

Integrate the delivery of the packages, which can be large EXE, DAT, or MSI files, with your existing organizational process. An existing process, such as Active Directory publishing using group policies, has an already established support structure and administration workflow. You can use group policies to deploy software to groups, organizational units, or individuals. See the following Microsoft article for details:

How to use Group Policy to remotely install software in Windows Server 2008 and in Windows Server 2003

Benefits

After the application package is delivered, application performance and availability are not subject to network dependencies.

Drawbacks

Package management is decentralized and updates to the applications will need to be performed on each machine. However, this can be managed with standard package deployment tools such as Microsoft System Center 2012 Configuration Manager (SCCM).

Application Registration

Application registration integrates the virtualized application package with the desktop operating system. Registration of a virtualized application creates:

- Shortcuts on the desktop and in the Start menu
- File-type, protocol, and object-type associations so that applications start automatically
- Entries in the Programs and Features applet of the Control Panel

The **Thinreg.exe** tool in ThinApp handles application registration. ThinApp MSI packages use Thinreg to perform application registration with the MSI installer, so registration always occurs for an application that is installed with an MSI. In addition, if you are not using an MSI installer, Thinreg can be run from a script or from the command line. **Thinreg.exe** can be local to the operating system or on a remote share. One example is to place the Thinreg tool in the netlogon share and call it from a Windows logon script. Administrators can run Thinreg against an entire directory of ThinApp packages by using an asterisk (*) as a wildcard character.

Application registration is not mandatory, as ThinApp packages can be started without registration. However, end users and administrators can benefit from registration.

ThinApp allows IT organizations to use either remote or local deployment, or to combine deployment methods and manage some applications centrally while deploying others to each end user. The same virtualized application package can be used with either deployment method. The application registration process performs the same function whether a package is local or remote.

The process of registering an application takes into account role-based access to applications.

Role-Based Access to Applications

The registration process can enumerate which users have access to an application package, so the administrator can register an entire directory of application packages. If an administrator uses a script that runs based on group membership, then the script will register only packages that are valid for certain groups or individuals. The following are two common methods of implementation using Active Directory.

Script-Based Registration

The Thinreg executable can be incorporated into an existing Windows logon script with standard methods such as .bat, WSH, KIX, or vbScript.

Note: Scripted Thinreg can only be used with EXE-based packages.

```
%logonserver%\netlogon\Thinreg.exe /Q \\company.com\applications\*.exe
```

The Application Registration Guide reviews this process in detail and can be used as a reference.

MSI-Based Registration

ThinApp package Delivery mechanisms often have an already-established support structure and administration workflow. You can use native Active Directory-based group policies to publish or assign an MSI package to groups, organizational units, or individuals. See the following VMware knowledge base article for details:

How to assign software to a specific group by using Group Policy in Windows Server 2003 (324750)

Appendix C: Optional Versus Required AppLinks

The load order for packages linked with AppLink is either alphabetical or in the order specified.

If a wildcard is used with Optional and there are multiple files that satisfy the condition, they are processed in alphabetical order. The following is an example of alphabetical load order of all EXE files:

```
OptionalAppLinks=PathToAppLinks\*.exe
```

In this example, the primary package searches for all files one folder deep in the specified AppLink path.

```
OptionalAppLinks=PathToAppLinks\*\*
```

Note: An error is generated if the directory specified has ALT files and a wildcard such as * is used. Using the wildcard means that AppLink processes all files (including ALT files), and this generates the error. To avoid this, you can combine **`*.exe`** and **`*.dat`**, as in the following example:

```
OptionalAppLinks=PathToAppLinks\*\*.exe; PathToAppLinks\*\*.dat
```

If different packages define conflicting settings, the package specified last is the one that takes precedence. For example, Package A defines the browser home page as A.com, and Package B defines the browser home page as B.com.

With the setting **`RequiredApplinks=A.dat; B.dat`**, the home page is **`B.com`**, and with **`RequiredApplinks=B.dat; A.dat`**, the home page is **`A.com`**.

It is recommended to link packages together in the order they would normally be captured. For example, a browser package is captured and linked to a browser plug-in package rather than the other way around. This way, the dependencies of the combined package are maintained.

This is very important in cases where changing the link order of the parent and child packages changes the behavior of the combined package.

The following VMware blog discusses AppLink in more detail: The power of AppLink.

Additional Resources

- ThinApp 5.2 trial
- ThinApp documentation
- ThinApp Community
- ThinApp technical papers
- ThinApp YouTube channel
- ThinApp Blog
- Horizon 7 product information
- Request a VMware Horizon 7 trial
- Horizon 7 documentation

About the Authors and Contributors

Gina Daly, Technical Marketing Manager, End-User-Computing Technical Marketing, VMware, updated this paper for ThinApp 5.2.

Tina de Benedictis, Senior Manager, End-User-Computing Technical Marketing, VMware, updated and enhanced this paper for ThinApp 4.7.

Aaron Black, Senior Product Manager for VMware Horizon, VMware, wrote the ThinApp 4.6 version of this paper while in the role of Technical Marketing Manager for ThinApp. His initial work formed the foundation for this updated paper.

The following individuals made significant contributions to the ThinApp 5.2 version of this paper:

- Jason Bassford, Technical Marketing Manager, End-User-Computing Technical Marketing, VMware
- Peter Bjork, Lead Specialist, EMEA End-User-Computing Practice, VMware

Thanks to Aaron Black, Coby Gurr, Vignesh Jayaraman, Mary Potapova, Sriram Nambakam, and John Domenichini for their contributions to the ThinApp 4.7 version.

To comment on this paper, contact VMware End-User-Computing Technical Marketing at euc_tech_content_feedback@vmware.com.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com